

Learning Generalizable Robot Skills for Dynamic and Interactive Tasks

Thesis

Saumya Saxena

CMU-RI-TR-25-84

August, 2025



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:

Prof. Oliver Kroemer, <i>Chair</i>	The Robotics Institute Carnegie Mellon University
Prof. Andrea Bajcsy	The Robotics Institute Carnegie Mellon University
Prof. Aaron Johnson	Dept. of Mechanical Engineering Carnegie Mellon University
Prof. Zico Kolter	Computer Science Department Carnegie Mellon University
Prof. Michael Posa	Dept. of Mechanical Engineering and Applied Mechanics (MEAM) University of Pennsylvania

*Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy*

© 2025 Saumya Saxena. All rights reserved.

Abstract

Recent years have seen increasing interest in developing robots that are capable of *lifelong* and *reliable* operations around humans in home and factory environments. Despite impressive recent progress towards long-horizon tasks such as laundry folding, current efforts are predominantly focused on quasi-static tasks in very structured settings. General-purpose robots that assist humans in daily tasks should be capable of performing a wider range of dynamic and dexterous tasks in unstructured environments, safely interact with humans and their environment, and effectively learn from and adapt to new experiences.

One promising approach to building robots capable of assistive lifelong operations is to endow them with three core capabilities: a metric, semantic, and temporal (memory) understanding of the world, the ability to perform long-horizon reasoning and planning, and the capacity to execute real-time, closed-loop policies that are dexterous, reactive, and safe in dynamic environments. Such an approach necessitates bridging a fundamental gap in the field. On one hand, traditional approaches based on classical control can provide strong guarantees for safety and optimality for real-time reactive control of dynamic tasks; however, they often rely on low-dimensional, fully observable states and often lack semantic awareness. In contrast, recent advances in vision-language-action models capture rich world semantics and generalize to novel settings, but are typically limited to quasi-static tasks and do not explicitly handle safety or adapt to dynamic changes in real time. The goal of this thesis work is to bridge this gap by exploring methods at the intersection of modern control theory, robot learning, and multimodal foundation models, with the goal of learning generalizable and safety-aware robot policies capable of performing complex, dynamic, and interactive tasks in unstructured environments.

Towards this goal, this thesis explores various approaches for leveraging strong structural, algorithmic and semantic priors that can enable both generalization as well as real-time reactive control of dynamic tasks. Our works (1; 2) focus on learning to perform dynamic pickup tasks using dynamic-Graph Neural Networks (GNNs) as structural priors and differentiable control algorithms (iLQR) as algorithmic priors for learning compositional skills that operate over multiple dynamic modes as well as generalize to complex environments and novel goals. (1) introduces a framework for learning differentiable optimal skills (LQR) for switching linear dynamical systems from expert demonstrations. The resulting control scheme predicts and accounts for discontinuities due to contact, reacts to unanticipated contact events and generalizes to novel goal conditions and skill compositions. (2) extends this framework to non-linear multibody interactive systems by learning stable locally linear dynamics models using dynamic-GNNs, which enables generalization to novel number of

objects and interactions unseen during training. MResT (3) combines the semantic reasoning capabilities of pre-trained frozen vision-language models (used at low frequency), with the adaptiveness of fine-tuned smaller networks (used at high frequency), to enable zero-shot generalization to semantic scene variations as well as real-time closed-loop control of precise and dynamic tasks. This work focuses on learning generalizable multi-task policies using multi-resolution sensing in table-top settings for short-horizon tasks. GraphEQA(4), considers the long-horizon task of embodied question answering in large unseen indoor environments. GraphEQA, in real-time, constructs a multi-modal memory comprising a real-time 3D metric-semantic scene graph and task-relevant images, for grounding a VLM-based hierarchical planner to perform situated and semantically-informed long-horizon exploration and planning. Finally, VLTSafe (5) combines the semantic safety reasoning capabilities of large pre-trained VLMs with safety guarantees of low-level reachability-based RL policies to enable safe control of dynamic tasks in cluttered environments. This approach uses a VLM for identifying relevant task- and safety constraints which inform pretrained parameterized reachability-based RL policies during execution.

This thesis takes a step towards the development of generalist embodied agents that integrate the semantic understanding and generalizability of multimodal foundation models with robust, closed-loop control policies that ensure efficiency, dexterity and safety and opens new avenues for future research in scalable, safe, and generalizable robot learning.

Acknowledgments

I would like to express my sincere gratitude to my thesis committee – Prof. Oliver Kroemer, Prof. Andrea Bajcsy, Prof. Zico Kolter, Prof. Aaron Johnson, and Prof. Michael Posa – for their guidance, thoughtful feedback, and support throughout my Ph.D. journey. I am especially grateful to my advisor, Prof. Oliver Kroemer, for his mentorship, patience, and encouragement. His guidance has been invaluable in shaping my growth as a researcher, and I am thankful for the freedom he gave me to pursue a wide range of research topics.

I am also deeply thankful to Prof. Andrea Bajcsy, whose energy and enthusiasm were always contagious. I left every meeting with renewed clarity and motivation. Her guidance was instrumental in shaping my understanding of safety in robotics and providing me with the resources and support to pursue this field.

I want to share my deep appreciation for my awesome and brilliant lab members of IAM lab – Mohit Sharma, Kevin Zhang, Jacky Liang, Tabitha Lee, Shivam Vats, Alex LaGrassa, Mark Lee, Zilin Si, Erin Zhang, Yunus Seker, Sarvesh Patil, Ami Sawhney, Xinyu Wang, and Janice Lee. From research discussions, paper writing to coding sessions and daily conversations, I learned a great deal from each of you. The supportive and collaborative environment in the lab made this journey not only more productive but also much more enjoyable.

I am grateful to my collaborators – Blake Buchanan, Chris Paxton, Jon Francis, Narunas Vaskevicius, Luigi Palmieri, Prof. Ralph Hollis, Prof. Nancy Pollard, Cornelia Bauer, Roberto Shu, Dr. Matt Travers, and Prof. Howie Choset – for the many insightful discussions and the opportunity to work on projects that not only broadened my perspective but also helped shape the trajectory of my research.

A special thank you goes to Suzanne Muth for her invaluable administrative support, which made navigating the countless logistical details of graduate school far easier. I am also thankful to Dr. Maureen Anderson and the CMU robotics outreach team. Participating in outreach was one of the most rewarding aspects of my time at CMU, giving me a sense of community, purpose, and fun every week.

To my friends, Micaela, Rhi, Pragna, Nico, Guillaume, Kati, Mark, Tanya, Ami, Alex, Shogo, Nikkhil, Sarah, Dimitri, and Cornelia, thank you for keeping me sane, and for the countless conversations and hangouts that carried me through the ups and downs of grad school.

To my dearest friends, Manas Singh and Harmanjit Singh, you are my family away from home. You always kept me grounded and believed in me even when I doubted myself. To my partner in crime, Blake – words will never be enough. Thank you for your love, patience, and constant support in everything I do. I continue to be amazed by your brilliance, humility, and kindness, and I feel lucky to have a close friend and confidant in you.

Finally, I would like to thank my family. To my brothers, Saurabh Saxena and Vaibhav Saxena, thank you for always being my sounding boards and for your con-

stant support and advice. To my mom, thank you for your endless love and encouragement. You have been my greatest source of inspiration – your life exemplifies resilience, positivity, and hard work. I strive every day to live up to the example you set. To my dad, I miss you deeply. Your belief in me, your pride in even my smallest accomplishments, and your constant encouragement shaped who I am today. I carry your lessons with me and strive each day to make you proud.

CONTENTS

Contents	vii
List of Tables	xii
List of Figures	xiv
1 Introduction	1
1.1 Motivation	1
1.1.1 Bridging the Generality-Reactivity gap	1
1.1.2 Challenges in dynamic manipulation	2
1.1.3 Lifelong robot operation in the real-world	2
1.2 Thesis Statement	3
1.3 Thesis Research Questions	3
1.4 Thesis Contributions	3
1.5 Thesis Conclusions	5
2 Learning Switching Linear Dynamical System Parameters for Differentiable Optimal Control	6
2.1 Introduction	6
2.2 Related Works	7
2.3 Problem Formulation	8
2.3.1 Modeling the Switching Linear Dynamical System	8
2.3.2 Learning the Switching Dynamics Model	10
2.3.3 Modeling the Differentiable Policy	11
2.3.4 Learning the Control Parameters	11
2.3.5 Execution	12
2.4 Experimental Results	14
2.4.1 Robot Experiments	17
2.5 Conclusion	18
2.6 Acknowledgments	18

3	Dynamic Inference on Graphs using Structured Transition Models	19
3.1	Introduction	19
3.2	Related Work	21
3.3	Method	22
3.3.1	Graph Inference Module	22
3.3.2	Forward Dynamics Module	24
3.3.3	Training	25
3.3.4	Execution	26
3.4	Experimental Results	26
3.5	Conclusion	31
3.6	Acknowledgments	31
4	Multi-Resolution Sensing for Real-Time Control with Vision-Language Models	32
4.1	Introduction	32
4.2	Related Work	34
4.3	Proposed Approach	35
4.3.1	Multi-Resolution Architecture	35
4.4	Experimental Setup	38
4.4.1	Environments	38
4.4.2	Baselines	39
4.5	Experimental Results	40
4.5.1	Comparison to Multi-Task Baselines	40
4.5.2	Additional Baseline Comparisons	41
4.5.3	Ablations	43
4.6	Conclusions and Limitations	44
4.7	Acknowledgments	44
5	VLTSafe: Vision-Language Guided Safe Dynamic Manipulation via Object-Centric Transformers	45
5.1	Introduction	46
5.2	Related work	47
5.3	Problem Formulation and the VLTSafe Method	48
5.3.1	<i>Offline</i> : Safe Policy Learning via Object-Centric Transformers & Reach-Avoid RL	49
5.3.2	<i>Online</i> : Adapting via Safety Constraints Inferred from Vision-Language Models	51
5.4	Simulation Experiments	52
5.4.1	Experimental Setup	52
5.4.2	How much does the transformer policy architecture influence task success?	53

5.4.3	How well can our approach generalize to varying degrees of test-time clutter?	55
5.4.4	How well does VLTSafe generalize to novel, test-time constraint compositions?	55
5.5	Real World Experiments	56
5.6	Conclusion	57
5.7	Limitations	57
6	GraphEQA: Using 3D Semantic Scene Graphs for Real-time Embodied Question Answering	58
6.1	Introduction	58
6.2	Preliminaries	60
6.2.1	Hierarchical 3D Metric-semantic Scene Graphs	60
6.2.2	2D Occupancy Mapping and Frontier Detection	61
6.3	Method	62
6.3.1	Problem Formulation	62
6.3.2	Scene Graph Construction and Enrichment	62
6.3.3	Task-relevant Visual Memory	63
6.3.4	Hierarchical Vision-Language Planner	63
6.4	Experimental Setup	65
6.4.1	Baselines and Ablations	65
6.5	Experimental Results	66
6.6	Conclusions	69
6.7	Limitations	69
7	Conclusions	71
8	Directions for future work	74
8.1	Lifelong Robotic Operation Through Structured Learning and Adaptation	74
8.2	Open-World Safety through Uncertainty-Aware and Human-in-the-Loop Control	75
I	Appendices	76
A	Appendix for Chapter 2	77
A.1	Training details	77
A.2	Task 1: 1-object pickup	77
A.2.1	Task 2: 2-object pickup	79
A.2.2	Task 3: Pickup task with ground interactions	80

B	Appendix for Chapter 3	82
B.1	Experiment Details	82
B.2	Additional results	82
B.2.1	Learned dynamic parameters and edge predictions	82
B.3	CONTROL USING LEARNED LOCALLY LINEAR MODELS	83
B.4	Cost Learning	85
C	Appendix for MResT	86
C.1	Environment Details	86
C.1.1	MT-Coarse Manipulation	86
C.1.2	MT-Precise Manipulation	86
C.1.3	MT-Dynamic Manipulation	87
C.2	Architecture Details	88
C.2.1	Training Details	89
C.2.2	Implementation Details	90
C.3	Additional Results	91
C.3.1	Additional Real-World Comparisons	91
C.3.2	Additional Ablations	92
D	Appendix for VLTSafe	94
D.1	Background	94
D.1.1	Hamilton-Jacobi Reachability Analysis	94
D.1.2	Reach-Avoid Reinforcement Learning (RARL)	94
D.1.3	Offline Parameterized Safe Policy Learning	95
D.2	Experimental details	95
D.2.1	Safety and Target Representation	96
D.2.2	Custom attention mask	97
D.2.3	Architecture and training details	97
D.3	VLM for Test-Time Safety and Target Specification	98
D.4	Real-world Experiments	99
D.4.1	Qualitative examples of VLM constraint selection	100
E	Appendix for GraphEQA	102
E.1	Habitat environment setup	102
E.2	HM-EQA and OpenEQA Datasets	102
E.3	Hydra 3D Scene Graphs	103
E.4	Prompting	103
E.4.1	GPT Prompt	103
E.4.2	Hierarchical Nature of 3DSGs and Planning	103
E.5	Baseline Performance on Task Categories	103
E.6	Additional Ablations:	104
E.7	Additional Quantitative results: Zero-shot reasoning for commonsense questions	104

E.8	Error Analysis of Competing Baselines	106
E.9	Structured Output	108
E.10	Additional Real-world Experiments in Home Environments	109
E.10.1	Home Environment (a)	109
E.10.2	Home Environment (b)	112
E.11	Additional Real-world Experiments in Office and Workshop Environments	114
E.11.1	Qualitative Analysis	114
E.11.2	Additional Real-world Experiments in an Office Environment .	116
E.12	Qualitative results	117
E.12.1	Exploration Efficiency and Trajectory Generation	117
E.12.2	Choosing object-nodes and frontiers semantically	118
Bibliography		124

LIST OF TABLES

2.1	RMSE of learned masses and learned goal configurations.	15
4.1	Task success comparison for multi-task baselines across all task domains.	40
4.2	Results for multi-spatial resolution experiments (Section 4.5.2). Here, $-$ implies that we remove this input from policy. Thus, π_{-Ih} implies that the policy only operates on third-person camera views and force-torque feedback.	41
4.3	Results for multi-temporal resolution experiments (Section 4.5.2). Here, both $\pi_{\text{low-res}}$ and $\pi_{\text{high-res}}$ are single-resolution approaches which run at 5 Hz and 20 Hz respectively, while ours is a multi-resolution approach.	41
4.4	Robustness experiment results, each cell shows <i>train/heldout</i> success rate (Section 4.5.2)	42
4.5	Mean (stdev) results (using 2 seeds) for multi-spatial resolution for real world tasks.	43
5.1	Policy Architecture vs. Performance. Comparison of our method against baselines in two training domains, <i>AllFragileConst</i> and <i>RandomConst</i>	53
6.1	Comparison to simulation baselines for HM-EQA and OpenEQA datasets: Success rate (%), average # of planning steps over successful trials, and L_τ average trajectory length over successful trials. Methods with a † indicate our implementations of that particular baseline.	67
6.2	Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.	68
A.1	Hyperparameters	77
C.1	Hyperparameters used for our architecture and model training.	89
C.2	Real-World results for using commonly used imitation learning (single-spatial resolution baselines) for Pickup task.	92
C.3	Additional Results for multi-temporal resolution experiments. As before, both $\pi_{\text{low-res}}$ and $\pi_{\text{high-res}}$ are single-resolution approaches which run at 5 Hz and 20 Hz respectively, while ours is a multi-resolution approach.	92

C.4	Results for using low-temporal resolutions for camera-inputs (5Hz) and high-temporal resolutions for force-torque only (75Hz).	93
D.1	Training hyperparameters	96
E.1	Success Rate (%) in simulation for task categories in the HM-EQA dataset, for Identification, Counting, Existence, State, and Location categories. Reported in terms of category successes / total number of category EQA tasks. [†] indicates our implementation of that baseline.	104
E.2	Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.	104
E.3	Additional baselines (Simulation): Success rate (%)	105
E.4	Error analysis (Simulation): Percentage %	106

LIST OF FIGURES

2.1	Training and execution pipeline.	9
2.2	Simulation qualitative results for three distinct tasks.	14
2.3	Real world demonstration.	16
2.4	(a) Success rate (b) Execution cost for the 3 Box2D tasks	16
3.1	Method overview.	23
3.2	Simulation results for pickup tasks and generalization scenarios. . . .	27
3.3	Simulation results for door opening task.	28
3.4	Simulation (Isaac Gym) and real world experiments with a Franka Arm performing dynamic manipulation task.	30
4.1	MResT method overview.	33
4.2	MResT training architecture.	36
4.3	Task settings in simulation and the real world.	37
4.4	Temporal resolution and robustness baselines used to compare our multi-resolution approach.	39
4.5	Example failure case for MT-Dynamic (Ballbot) task.	42
4.6	<i>Left:</i> Ablation results (see Section 4.5.3). <i>Right:</i> Robustness result for real-world pickup.	42
5.1	VLTSafe: Method Overview	50
5.2	VLTSafe System Architecture	51
5.3	Results: Generalization to varying Degrees of Clutter.	54
5.4	Results: Generalization to novel constraint compositions.	56
5.5	Real-world deployment: Zero-shot sim-to-real transfer of a policy trained in simulation	57
6.1	GraphEQA: Method Overview.	60
6.2	Overall GraphEQA architecture.	61
6.3	VLM Planner Architecture.	63
6.4	Deploying GraphEQA on the Hello Robot Stretch RE2 platform in two unique home environments	69

C.1	Real world environment setup and example objects used in the experiments.	88
D.1	Custom attention mask	96
D.2	Attention block	97
D.3	Real-world deployment: Zero-shot sim-to-real transfer of a policy trained in simulation	100
D.4	Qualitative results for the VLM selecting safety constraint types and target regions.	100
E.1	An example of a false positive case.	107
E.2	An example of a false negative case.	107
E.3	Additional Real-world Experiments in Home Environment (a) Question 1	109
E.4	Additional Real-world Experiments in Home Environment (a) Question 2	110
E.5	Additional Real-world Experiments in Home Environment (a) Question 3	110
E.6	Additional Real-world Experiments in Home Environment (a) Question 4	111
E.7	Additional Real-world Experiments in Home Environment (a) Question 5	111
E.8	Additional Real-world Experiments in Home Environment (b) Question 1	112
E.9	Additional Real-world Experiments in Home Environment (b) Question 2	112
E.10	Additional Real-world Experiments in Home Environment (b) Question 3	113
E.11	Task-relevant image selection	114
E.12	Additional Real-world Experiments in workshop environment.	116
E.13	Additional Real-world Experiments in Office environment.	117
E.14	Qualitative comparison of exploration trajectories of ExploreEQA vs GraphEQA.	118

1

INTRODUCTION

1.1 Motivation

Robotic systems have the potential to greatly improve human quality of life by assisting in hazardous and caregiving settings. In hospitals and homes, assistive robots can reduce the burden on caregivers by helping with patient monitoring, mobility, and routine tasks for the elderly. In industrial settings such as mines, nuclear facilities, and chemical plants, robots can take on physically demanding or high-risk tasks, improving both safety and operational efficiency. This thesis takes a step towards the long-term goal of building general-purpose robots capable of lifelong and reliable operations in open-world settings and capable of safe collaboration with humans in assistive and caregiving roles.

1.1.1 Bridging the Generality-Reactivity gap

Despite impressive progress towards developing generalist embodied agents (6; 7; 8) that can perform long-horizon tasks – such as folding laundry, loading dishwashers, or assembling simple meals – these approaches are often limited to quasi-static structured settings. General-purpose assistive robots should be capable of performing a wider range of dynamic and dexterous tasks in unstructured environments, safely interact with humans and their environment, and continuously learn from new experiences. This requires bridging this gap between generalist quasi-static policies and policies for performing dynamic and dexterous tasks in open-world environments. This thesis explores methods at the intersection of modern control theory, robot learning, and multimodal foundation models, with the goal of learning generalizable and safety-aware robot policies for performing complex, dynamic, and interactive tasks in unstructured environments.

1.1.2 Challenges in dynamic manipulation

We need robots to be agile so that they can seamlessly perform tasks such as picking up a bottle in a single sweeping motion, pulling a magazine from under a stack of books, or quickly catching a falling object. In open-world, dynamic human environments, such agility is essential not only for efficiency but also for robust failure recovery. These tasks, however, are highly challenging: they involve complex dynamics, including velocities, inertia, and contacts, as well as multi-body interactions with significant momentum transfer. Even small errors can lead to catastrophic failures, for example, a single jerky motion in a stacking task can trigger multiple unintended interactions and collapse the entire stack. In cluttered settings, the sheer number of potential interactions further amplifies these risks. Crucially, such skills must generalize across objects, tasks, and environments, where both dynamics and semantics vary widely. Collecting data for these tasks is significantly more difficult than in quasi-static settings, making it imperative to develop methods that are data-efficient and capable of learning in low-data regimes. To meet these challenges, policies must be both reactive, responding to high-frequency sensory feedback and contact events, and predictive, reasoning about the long-term consequences of actions and interactions. They must also be able to focus attention on the most critical objects and interactions to avoid distraction and mitigate risk. Ultimately, attention, foresight, and reactivity are the key capabilities a policy must possess to successfully perform dynamic manipulation in open-world settings. This thesis investigates these core components with a focus on learning generalizable skills for dynamic interactive tasks.

1.1.3 Lifelong robot operation in the real-world

Achieving the broader goal of lifelong robotic operation in dynamic, unstructured human environments demands a more general and hierarchical framework. This framework can be conceptualized as consisting of three key components: (1) a compact and effective memory that encodes metric, semantic, and temporal information; (2) a high-level planning and reasoning module capable of long-horizon task decomposition and semantic understanding; and (3) a robust low-level control policy that integrates multisensory inputs to ensure dexterity, reactivity, and safety during task execution. Each of these components must be deliberately structured to support interpretability, generalization, and lifelong adaptability. For instance, memory representations should be compact and modular, capturing both task-specific and task-agnostic knowledge in a way that can be efficiently updated as the robot explores and interacts with its environment. The planning and reasoning module should leverage this memory, alongside high-level task instructions, to decompose complex tasks into compositions of pre-trained/hand-designed skills from the robot’s skill library. Finally, low-level control skills should be designed to prioritize reactivity, dexterity, efficiency, and safety. This thesis investigates core components of such an architecture, with a focus on how embedding structural, algorithmic, and semantic priors

into robot learning systems can facilitate long-horizon reasoning, safe and dynamic execution, and generalization to novel domains.

1.2 Thesis Statement

We now introduce the statement of this thesis that frames the broader vision of this body of work.

The objective of this thesis is to develop robot learning algorithms that leverage attention, reactivity, and foresight to learn generalizable and reliable skills for dynamic and interactive tasks.

1.3 Thesis Research Questions

This thesis work aims to address the following research questions that entail the above thesis statement.

- R1. How can we utilize algorithmic priors to enable efficient learning of *predictive and reactive* skills for dynamic and interactive tasks?
- R2. How can we ensure *attention* to relevant objects and interactions in unstructured environments and enable generalization to arbitrary objects and interactions?
- R3. How can we leverage multimodal sensing to enable language-conditioning and *semantic generalization* as well as real-time reactive control of dynamic and precise tasks?
- R4. How can learn skills that ensure *long-term safety* for dynamic manipulation tasks while enabling *semantic safety* reasoning and generalization in novel cluttered environments?
- R5. For performing long-horizon navigation tasks in open-world environments, how can we construct a real-time compact multimodal actionable memory that enables in-context planning and exploration?

1.4 Thesis Contributions

- C1. *Learning Reactive and Predictive Differentiable Controllers for Switching Linear Dynamical Models (1)*: A framework for learning a switching linear dynamical model with contacts encoded in switching conditions as a close approximation of system dynamics. Using discrete-time LQR as the differentiable policy class for data-efficient learning of control to develop a control strategy that operates

over multiple dynamical modes and takes into account discontinuities due to contact. In addition to being sample efficient, this policy class provides a feedback control scheme that captures expert behavior in the learned cost/value functions which can be used to generalize to other scenarios. In particular, using cost matrices learned from a single scenario, we show generalization to different goal conditions while achieving the same overall behavior. We show that, in addition to *predicting* interactions with the environment, our policy effectively *reacts* to inaccurate predictions such as unanticipated contacts.

→ Chapter 2, *Research Question: R1*

- C2. *Dynamic Inference on Graphs using Structured Transition Models (2)*: A framework for learning the dynamics of non-linear interactive systems by learning stable locally-linear dynamics models using graph neural networks (GNNs). GNNs provide a principled way of learning the dynamics of large interactive systems by incorporating a strong structural prior in the learned dynamics. This work incorporates learning a dynamic graph structure that encodes contacts by adding and removing edges from the graph as contact is made and broken which allows for stronger generalization of the learned interaction dynamics to more objects/interactions than seen during training.

→ Chapter 3, *Research Questions: R1, R2*

- C3. **MResT** - *Multi-Resolution Sensing for Real-Time Control with Vision-Language Models (3)*: A framework for learning generalizable language-conditioned policies for real-time control of precise and dynamic tasks by combining sensing at different resolutions using networks of varying capacities. We demonstrate, that tasks such as dynamically picking up an object require both global information from turret cameras as well as local information from in-hand cameras and force-torque feedback from the wrist sensors for successful task completion. Moreover, using large pretrained vision language models to extract high-level information from third-person cameras allows generalization of the learned skills to variations in the semantics of the scene.

→ Chapter 4, *Research Questions: R2, R3*

- C4. **VLTSafe** - *Vision-Language Guided Safe Dynamic Manipulation via Object-Centric Transformers (5)*: A framework for safe dynamic manipulation that leverages vision-language models to translate semantic safety concepts into geometric constraints, and object-centric transformers to learn generalizable low-level safe policies. We optimize a reach-avoid reinforcement learning objective to train a parameterized policy that reasons about diverse combinations of task and safety constraints. During inference a VLM identifies relevant geometric constraints to parameterize the learned policy enabling zero-shot generalization to highly cluttered scenes and novel constraint compositions.

→ Chapter 5, *Research Questions: R1, R2, R4*

- C5. **GraphEQA** - *Using 3D Semantic Scene Graphs for Real-time Embodied Question Answering (4)*: We address the problem of Embodied Question Answering (EQA) in unseen environments by introducing a novel approach that leverages real-time 3D metric-semantic scene graphs (3DSGs) and task-relevant images as multi-modal memory to ground vision-language-based planning agents. We employ a hierarchical planning approach that exploits the hierarchical nature of 3DSGs for structured planning and semantics-guided exploration. We demonstrate that this approach outperforms key baselines by completing EQA tasks with higher success rates and fewer planning steps via experiments in simulation and the real world.

→ Chapter 6, *Research Questions: R2, R5*

1.5 Thesis Conclusions

As a consequence of these contributions, we present the conclusions of this thesis. These conclusions reiterate the thesis statement – attention to relevant features, foresight and long-horizon reasoning, and reactivity (high-frequency sensing/control) are essential components for reliable robot operations in open-world environments.

- T1. Learning differentiable LQR policies for switching dynamical systems enables predictive and reactive control for dynamic and interactive manipulation tasks as well as generalization to novel goal conditions.

→ *Contribution: C1*

- T2. Attention to relevant objects and interactions via Dynamic-GNNs enables robustness to distractors and generalization to novel interactions.

→ *Contribution: C2*

- T3. MResT: Attention to sensing at different resolutions and semantic reasoning via pretrained VLMs enables semantic generalization as well as real-time reactive control of dynamic and precise tasks.

→ *Contribution: C3*

- T4. VLTSafe: Attention to relevant objects and near-term interactions, long-term safety considerations (foresight) via reach-avoid RL, and semantic safety reasoning using VLMs, enables semantic generalization as well as predictive and reactive safe control of dynamic manipulation tasks.

→ *Contribution: C4*

- T5. GraphEQA: Multimodal memory with attention to relevant images and VLM-based planning (foresight) enables performing long-horizon embodied tasks in unseen environments.

→ *Contribution: C5*

2

LEARNING SWITCHING LINEAR DYNAMICAL SYSTEM PARAMETERS FOR DIFFERENTIABLE OPTIMAL CONTROL

Humans leverage the dynamics of the environment and their own bodies to accomplish challenging tasks such as grasping an object while walking past it or pushing off a wall to turn a corner. Such tasks often involve switching dynamics as the robot makes and breaks contact. Learning these dynamics is a challenging problem and prone to model inaccuracies, especially near contact regions. In this work, we present a framework for learning composite dynamical behaviors from expert demonstrations. We learn a switching linear dynamical model with contacts encoded in switching conditions as a close approximation of our system dynamics. We then use discrete-time LQR as the differentiable policy class for data-efficient learning of control to develop a control strategy that operates over multiple dynamical modes and takes into account discontinuities due to contact. In addition to *predicting* interactions with the environment, our policy effectively *reacts* to inaccurate predictions such as unanticipated contacts. Through simulation and real world experiments, we demonstrate generalization of learned behaviors to different scenarios and robustness to model inaccuracies during execution.

2.1 Introduction

Everyday human tasks involve intricate interactions with the environment, manifesting in changing dynamics and impacts. For example, to pick up a bottle in one smooth motion while walking past it, a human first slows down, comes in contact, and then applies an appropriate force to pick it up without toppling it over. Modeling the dynamics for such tasks is a challenging problem. Even small inaccuracies in the learned model, especially near the contact regions, can cause catastrophic failures and unstable behaviour. One approach to prevent such problems is to not rely on

the learned models entirely (9; 10), rather use them as close approximations of the system behavior, and at the same time design/learn a controller that is both robust to the model inaccuracies and stable under execution noise.

In this work, we develop a control strategy that operates over multiple dynamical modes, takes into account discontinuities due to contact, and is robust to inaccuracies in the learned dynamics model. Our approach focuses on learning the dynamic behavior of an expert and then generalizing that behavior to unseen scenarios.

Systems for which the dynamics change discontinuously, usually after contact is made or broken, can be modeled as *switching linear dynamical systems* (SLDS) (11; 12; 13; 14; 15; 16). The set of discrete dynamics in a SLDS are governed by hidden states called *modes*. We model impacts as conditions governing the switch between these modes, which incorporates discontinuities into our dynamics model. We use system identification to learn the parameters of the SLDS including the impact dynamics and mode prediction function.

Taking inspiration from recent approaches that add prior structure to policy architecture (17; 18; 19) for data-efficient learning of control, we use discrete-time LQR as our differentiable policy class. In addition to being sample efficient, this policy class provides a feedback control scheme that captures expert behavior in the learned cost/value functions which can be used to generalize to other scenarios. For our complex task modeled as a SLDS, we use a single LQR policy and parameterize it using a different cost function and goal condition for each mode. Using cost matrices learned from a single scenario, we can generalize to different goal conditions while achieving the same overall behavior.

Forward propagating the control using the learned model allows us to predict mode switches and contacts by extension. To make the controller robust to inaccurate mode predictions, we combine our method with a reactive control scheme (20), to switch modes upon *observing* inaccurate predictions of state transitions. We use model predictive control during execution to replan using the observed modes.

The contributions of this paper are twofold: 1) a method for modeling and learning switching contact-based linear dynamics and using it for efficiently learning a differentiable feedback controller that operates over multiple dynamical modes, takes into account discontinuities due to contact, and generalizes it to unseen scenarios, 2) a control scheme that is robust to inaccuracies in the learned dynamics model and during execution can predict, as well as react to, unanticipated dynamic contact events. An overview of the method and results can be found in the supplementary video ([link](#)).

2.2 Related Works

Contacts and changing dynamics are ubiquitous in manipulation (21) and collaborative tasks. Model-based control approaches for learning contact-rich manipula-

tion have been proposed (22), (23), but their asymptotic performance suffers due to model-bias. To alleviate this problem, other approaches combine model-based and model-free methods (9; 24). Our approach relies on using a robust control scheme (25) to deal with inaccuracies in the learned dynamics model.

Modeling tasks with non-stationary dynamics as switching linear dynamical systems (SLDS) is a well known problem. There are approaches that do (15; 16) and do not (11; 12; 13; 14) model discontinuous mode transitions. Some other approaches for modelling contacts and learning contact based dynamics include (26; 27; 28; 29). Our approach for modeling contact as mode switches compares best with (30), where the mapping from pre-impact velocity to post-impact velocity is learned. Methods such as (11; 31) learn controllers for each mode separately, and switch between controllers using the learned predictive guard conditions. Such methods are susceptible to failures due to inaccuracies in these learned conditions. Methods such as eMO-SAIC (20), on the other hand, do not learn a predictive model for mode switching, but switch modes and control reactively upon observed changes in the effects of actions. In our method, we combine the advantages of both the predictive and reactive control schemes to develop a robust controller that acts over multiple discrete modes and contact.

Model-free approaches (32; 33; 34) obviate the need for learning the dynamics, but face challenges in terms of sample complexity. Recent approaches to alleviate these challenges propose adding prior structure to the policy architecture (17; 18; 19). We take inspiration from this work and use discrete-time LQR as our differentiable policy class.

2.3 Problem Formulation

In this section we first elaborate on our method for modeling and learning parameters of the switching dynamical system using system identification. We then model the controller and learn the control parameters using imitation learning. Finally we introduce our robust control scheme for executing the learned policy.

2.3.1 Modeling the Switching Linear Dynamical System

Graphical model for our switching dynamical system is shown in the second panel in Fig. 2.1(a). State of the system at time t is given by $\mathbf{x}_t = [\mathbf{q}_t, \dot{\mathbf{q}}_t]$ where $\mathbf{x}_t \in \mathbb{R}^n$, \mathbf{q}_t , $\dot{\mathbf{q}}_t$ are configuration and velocities respectively. State is composed of state of the robot and states of objects of interest in the environment. At every state \mathbf{x}_t the robot takes an action $\mathbf{u}_t \in \mathbb{R}^m$ and transitions to state \mathbf{x}_{t+1} . The transition dynamics and the control parameters at every time step are dependent on the discrete *mode* ρ_t , which is a hidden variable.

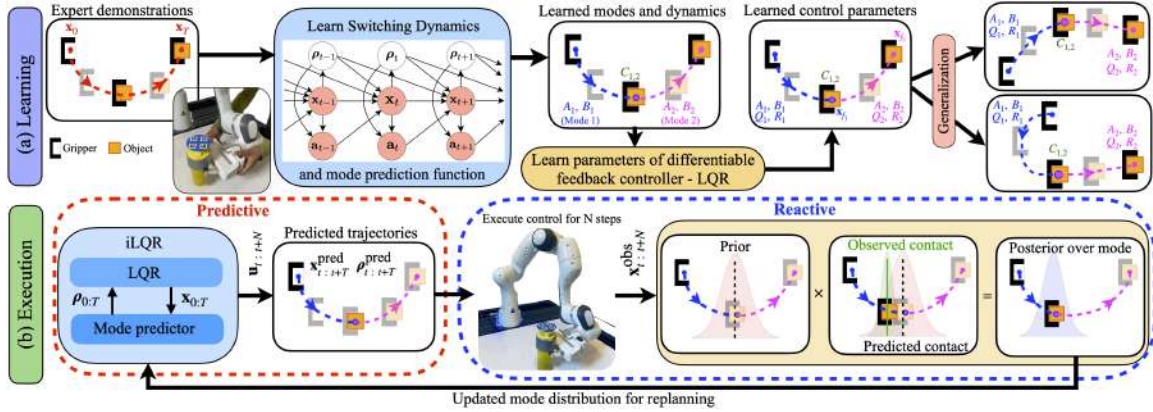


Figure 2.1: Training and execution pipeline (a) Given a set of expert demonstrations, learn the parameters of a switching linear dynamical system (transition dynamics, contact dynamics and mode prediction function) using system identification. Using the learned dynamics parameters and the expert demonstrations, learn a differentiable feedback controller (LQR), parameterized using a different cost function and goal condition for each mode, using imitation learning. Learning the cost matrices allows us to capture the behavior of the expert in each mode, thus enabling the learned controller to generalize to unseen scenarios. (b) During execution, we use iLQR to iteratively evaluate the modes along a trajectory, and use these modes and hence the corresponding dynamics, impact and cost parameters to solve the LQR problem. To make the controller robust to inaccuracies in the learned model, we use MPC to combine our predictive controller with a reactive control scheme. The predictive part adapts the control in anticipation of mode switches, and the reactive part observes unanticipated mode transitions, updates the current mode and replans.

Mode Prediction

We assume that the current mode ρ_t is only dependent upon the current state \mathbf{x}_t . We call the function for mode prediction a *classifier* $\rho_t = M(\mathbf{x}_t)$. The classifier is modeled as a fully-connected neural network that takes as input current state \mathbf{x}_t and outputs a one-hot vector over modes. For a picking task the state \mathbf{x}_t will include the position and orientation of the object we want to pick up. The classifier can learn to predict modes that depend upon more general conditions such as relative position between the gripper and the object, as used in our model. We always set the number of modes in the model to be greater or equal to the actual number of modes required by the task. The number of learned modes can be less than or equal to the maximum value specified.

Transition and Interaction dynamics

The transition dynamics are modeled as follows:

$$\mathbf{x}_{t+1} = \mathbf{A}_{\rho_t} \mathbf{C}_{\rho_t, \rho_{t-1}} \mathbf{x}_t + \mathbf{B}_{\rho_t} \mathbf{u}_t \quad (2.1)$$

where $\mathbf{A}_{\boldsymbol{\rho}_t} \in \mathbb{R}^{n \times n}$ and $\mathbf{B}_{\boldsymbol{\rho}_t} \in \mathbb{R}^{n \times m}$ correspond to the discrete mode $\boldsymbol{\rho}_t$ at time t . Thus, for a set of N modes, there will be N such transition matrices that we need to learn. The matrix $\mathbf{C}_{\boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}} \in \mathbb{R}^{n \times n}$ represents the impact or interaction dynamics which depends upon the mode switch between the current and previous time step. For example, in case of a pushing task with two modes, the first mode is when the robot is in free motion and the second mode is when it is in contact with the light object and is pushing it. An impact occurs between the robot and the object at the instant when contact is made, that is, when the mode *switches* from mode 1 to 2. Upon impact, the velocity of a system changes instantaneously while the position remains the same, which can be modeled as momentum transfer between the robot and the object. This is the source of discontinuity in the dynamics which we model here explicitly.

We take inspiration from approaches that model contact as a mapping from pre-impact to post-impact velocity (30). Consider a simple system where the gripper and object states are given as $\mathbf{x}_t^g = [\mathbf{q}_t^g, \dot{\mathbf{q}}_t^g]$ and $\mathbf{x}_t^o = [\mathbf{q}_t^o, \dot{\mathbf{q}}_t^o]$, respectively. The impact dynamics for this system can be written as

$$\begin{bmatrix} \mathbf{q}_{t+}^g \\ \dot{\mathbf{q}}_{t+}^g \\ \mathbf{q}_{t+}^o \\ \dot{\mathbf{q}}_{t+}^o \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{m_g - em_o}{m_g + m_o} & 0 & \frac{(1+e)m_o}{m_g + m_o} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{(1+e)m_g}{m_g + m_o} & 0 & \frac{m_o - em_g}{m_g + m_o} \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} \mathbf{q}_{t-}^g \\ \dot{\mathbf{q}}_{t-}^g \\ \mathbf{q}_{t-}^o \\ \dot{\mathbf{q}}_{t-}^o \end{bmatrix},$$

where e is the coefficient of restitution between the gripper and the object, m_g and m_o are the mass matrices for the gripper and the object respectively, and $\mathbf{q}_{t-}, \dot{\mathbf{q}}_{t-}$ and $\mathbf{q}_{t+}, \dot{\mathbf{q}}_{t+}$ are the position and velocity before and after impact respectively. We observe that some properties of this impact/interaction matrix \mathbf{C} make it easier to learn from demonstration and apply it in a wide range of applicable scenarios. First, \mathbf{C} is only dependent on the system properties and not on the velocity or position. Thus, we can parameterize and learn it using expert demonstrations and then use it in different scenarios. Second, we need only learn the parameters corresponding to changes in velocity because the position remains the same during impact. Finally, even though we model a different impact matrix for each possible mode switch, we end up learning impact matrices only for the mode switches that are observed during demonstrations, pruning the number of impact matrices that need to be learned. To ensure that the impact only occurs upon a mode *switch*, we write the impact dynamics as $\mathbf{C}_{\boldsymbol{\rho}_{t-1}, \boldsymbol{\rho}_t} = \boldsymbol{\rho}_{t-1}^\top \boldsymbol{\rho}_t \mathbf{I} + (1 - \boldsymbol{\rho}_{t-1}^\top \boldsymbol{\rho}_t) \mathbf{C}$ because when $\boldsymbol{\rho}_{t-1} = \boldsymbol{\rho}_t$, $\mathbf{C}_{\boldsymbol{\rho}_{t-1}, \boldsymbol{\rho}_t} = \mathbf{I}$ and when $\boldsymbol{\rho}_{t-1} \neq \boldsymbol{\rho}_t$, $\mathbf{C}_{\boldsymbol{\rho}_{t-1}, \boldsymbol{\rho}_t} = \mathbf{C}$ where \mathbf{C} is learned from demonstrations.

2.3.2 Learning the Switching Dynamics Model

Having defined the structure of the dynamical system, we move on to learning the model parameters from expert demonstrations using system identification. The

expert demonstrations are given as a set of trajectories $\{\mathbf{x}_t^*, \mathbf{u}_t^*, \mathbf{x}_{t+1}^*\}_{t=\{0,\dots,T-1\}}$ where T is the total length of the trajectory. The system identification cost is written as:

$$\begin{aligned}\boldsymbol{\rho}_t &= M(\mathbf{x}_t^*), \quad \mathbf{x}_{t+1}^{\text{pred}} = \mathbf{A}_{\boldsymbol{\rho}_t} \mathbf{C}_{\boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}} \mathbf{x}_t^* + \mathbf{B}_{\boldsymbol{\rho}_t} \mathbf{u}_t^* \\ \mathcal{L}_{\text{SysID}} &= \|\mathbf{x}_{1:T}^* - \mathbf{x}_{1:T}^{\text{pred}}\|_2.\end{aligned}$$

Taking gradient with respect to the above cost we together learn $\mathbf{A}_{\boldsymbol{\rho}_t}, \mathbf{B}_{\boldsymbol{\rho}_t}, \mathbf{C}_{\boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}}$, and the classifier $M(\cdot)$.

2.3.3 Modeling the Differentiable Policy

Having learned the system dynamic parameters, we shift our focus to modeling a controller that can perform the task demonstrated by the expert. More importantly, the controller should 1) robustly perform the task, 2) match the behavior of the expert through the learned cost function, such as slowing down before grasping, 3) generalize to other task instances that use the same objects. We design our controller as a discrete-time linear quadratic regulator (LQR), a differentiable feedback controller that can be represented as the solution to the following optimization problem:

$$\begin{aligned}\min_{\mathbf{u}_t} \quad & \sum_{t=0}^T \left((\mathbf{x}_t - \mathbf{x}_{f_{\boldsymbol{\rho}_t}})^\top \mathbf{Q}_{\boldsymbol{\rho}_t} (\mathbf{x}_t - \mathbf{x}_{f_{\boldsymbol{\rho}_t}}) + \mathbf{u}_t^\top \mathbf{R}_{\boldsymbol{\rho}_t} \mathbf{u}_t \right) \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = \mathbf{A}_{\boldsymbol{\rho}_t} \mathbf{C}_{\boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}} \mathbf{x}_t + \mathbf{B}_{\boldsymbol{\rho}_t} \mathbf{u}_t, \quad i = 1, \dots, T \\ & \mathbf{x}_0 = \mathbf{x}_{\text{init}}\end{aligned}\tag{2.2}$$

where $\mathbf{x}_{f_{\boldsymbol{\rho}_t}}, \mathbf{Q}_{\boldsymbol{\rho}_t}, \mathbf{R}_{\boldsymbol{\rho}_t}$ correspond to the goal configuration and cost matrices for mode $\boldsymbol{\rho}_t$. This formulation allows us to capture the behavior of the system in each mode using the cost matrices \mathbf{Q} and \mathbf{R} for a corresponding goal location \mathbf{x}_f . It is important to note that the behavior over the entire task is governed by the sequence of behaviors of the learned modes. Thus, once learned from expert demonstrations, as illustrated in Section 2.3.2, the *sequence* of modes should be kept fixed to successfully complete the task.

Using LQR feedback controller enables optimal responses to anticipated changes in dynamics and impact conditions in the trajectory. By learning cost matrices, the agent is essentially learning a value function or reward structure around each goal configuration. Using learned cost matrices allows the controller to generalize to different sub-goal configurations along the trajectory. Keeping the sequence of modes, and thus the *behavior* along the trajectory, fixed, we can shift around the learned sub-goal locations (\mathbf{x}_f) to new desired locations and still attain the same overall behavior.

2.3.4 Learning the Control Parameters

As illustrated in Section 2.3.2, for each expert demonstration, we have already learned the mode prediction $\boldsymbol{\rho}_t = M(\mathbf{x}_t^*)$ and the corresponding dynamic parameters

$\mathbf{A}_{\rho_t}, \mathbf{B}_{\rho_t}, \mathbf{C}_{\rho_t, \rho_{t-1}}$ for each time step. Now, we use imitation learning to learn the control parameters $\mathbf{x}_{f_\rho}, \mathbf{Q}_\rho$ and \mathbf{R}_ρ . For each expert initial condition $\{\mathbf{x}_0^*\}_{t=\{0, \dots, T\}}$, using the corresponding learned dynamic parameters and modes, we solve the optimization problem in (2.2) using discrete-time Riccati equations (?). Let the solution to the LQR problem for each initial state \mathbf{x}_0^* be given as

$$\mathbf{x}_{0:T}^{\text{LQR}}, \mathbf{u}_{0:T-1}^{\text{LQR}} = \text{LQR}(\mathbf{x}_0^*, \mathbf{A}_{\rho_t}, \mathbf{B}_{\rho_t}, \mathbf{C}_{\rho_t, \rho_{t-1}}, \mathbf{Q}_{\rho_t}, \mathbf{R}_{\rho_t}, \mathbf{x}_{f_{\rho_t}}).$$

The imitation cost can then be written as

$$\mathcal{L}_{\text{Imitation}} = \|\mathbf{x}_{0:T}^* - \mathbf{x}_{0:T}^{\text{LQR}}\|_2 + \|\mathbf{u}_{0:T-1}^* - \mathbf{u}_{0:T-1}^{\text{LQR}}\|_2.$$

The differentiability of the LQR controller allows us to learn the goal configurations \mathbf{x}_{f_ρ} and cost functions $\mathbf{Q}_\rho, \mathbf{R}_\rho$ corresponding to each mode by minimizing the above cost.

2.3.5 Execution

We have shown how we learn the dynamics model and controller from expert demonstrations. Now we move on to how we can solve similar tasks with new initial configurations of the robot and objects in the environment using these learned parameters. We also show how the controller accounts for inaccuracies in the learned dynamics model.

Iterative Linear Quadratic Regulator

Algorithm 1 Iterative LQR

(i) represents the iteration number
 T is the length of the trajectory
 $j, k \in \{1, \dots, N\}$ where N is the number of modes
Given: Initial state \mathbf{x}_{init} . Initialize modes $\rho_{0:T}$
Parameters: $M(\cdot), \mathbf{A}_{\rho_j}, \mathbf{B}_{\rho_j}, \mathbf{C}_{\rho_j, \rho_k}, \mathbf{x}_{f_{\rho_j}}, \mathbf{Q}_{\rho_j}, \mathbf{R}_{\rho_j}$
procedure ILQR(Parameters)
 while not converged **do**
 $\mathbf{x}_{0:T}^{(i)}, \mathbf{K}_{0:T-1} \leftarrow \text{LQR}(\mathbf{A}_{\rho_t^{(i)}}, \mathbf{B}_{\rho_t^{(i)}}, \mathbf{C}_{\rho_t^{(i)}, \rho_{t-1}^{(i)}}, \mathbf{x}_{f_{\rho_t^{(i)}}}, \mathbf{Q}_{\rho_t^{(i)}}, \mathbf{R}_{\rho_t^{(i)}})$
 $\rho_t^{(i+1)} = M(\mathbf{x}_t^{(i)})$ for all t
 converged $\leftarrow \|\mathbf{x}_{0:T}^{(i+1)} - \mathbf{x}_{0:T}^{(i)}\|_2 + \|\mathbf{u}_{0:T-1}^{(i+1)} - \mathbf{u}_{0:T-1}^{(i)}\|_2$ is small
 end while
end procedure

We formulate the problem of evaluating the control sequence for tasks involving multiple discrete dynamical modes as an iterative LQR problem. The approach is

very similar to a standard iLQR problem where, starting with an initial trajectory, we linearize the dynamics around the trajectory and use these dynamics to solve the optimal LQR problem iteratively. In our approach, as illustrated in Algorithm 3, at every iteration, we evaluate the mode for each step of the trajectory and populate the dynamics, impact and cost matrices accordingly to iteratively solve the LQR problem. It is important to note here that, from the expert demonstrations, we learned the cost matrices that capture the behavior of the system while achieving goal states $\mathbf{x}_{f\rho_j}$. By substituting $\mathbf{x}_{f\rho_j}$ with new desired goal states, we can generalize to different initial and goal configurations for a task with the same sequence of modes. A high-level planner can be used to rearrange the modes to generalize to similar tasks with different mode sequences, but this has been left for future work.

Model Predictive Control

Algorithm 2 Model Predictive Control

```

 $T_{LQR}$  is time horizon for execution before replanning
 $N_{MPC} = T/T_{LQR}$ 
 $T$  is the length of the trajectory
 $t$  is the current time step
 $t \leftarrow 0$ 
for  $n$  in  $N_{mpc}$  do
   $\mathbf{K}_{t:T} \leftarrow \text{iLQR}(\mathbf{x}_t, \mathbf{A}_\rho, \mathbf{B}_\rho, \mathbf{C}_{\rho,\rho}, \mathbf{R}_\rho, \mathbf{Q}_\rho, \mathbf{x}_{f\rho, T-t})$ 
  for  $i$  in  $T_{LQR}$  do
     $\rho_t^{post} \leftarrow \text{Posterior}(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{a}_{t-1})$   $\triangleright (3.3.4)$ 
     $\mathbf{a}_t \leftarrow \mathbf{K}_t(\mathbf{x}_t - \mathbf{x}_{f\rho_t}^{post})$ 
     $\mathbf{x}_{t+1} \leftarrow \text{step}(\mathbf{a}_t)$ 
     $t \leftarrow t_{t+1}$ 
  end for
end for

```

Since LQR is a feedback controller, it is robust to small disturbances and noise that are encountered during execution. However, our controller should be robust to inaccuracies in our learned model, especially the learned mode predictor $M(\mathbf{x}_t)$. Consider a scenario where for a grasping task, the classifier learns that contact occurs when the measured relative position between the gripper and the object is less than some threshold value. During execution, contact could potentially occur a little before, or a little after, this threshold is crossed. If the threshold is a little too large and the next mode requires the robot to move away from the object location, the robot might start moving away before actually grasping the object. To make the system robust to such scenarios, we use model predictive control (MPC), Algorithm 4, to

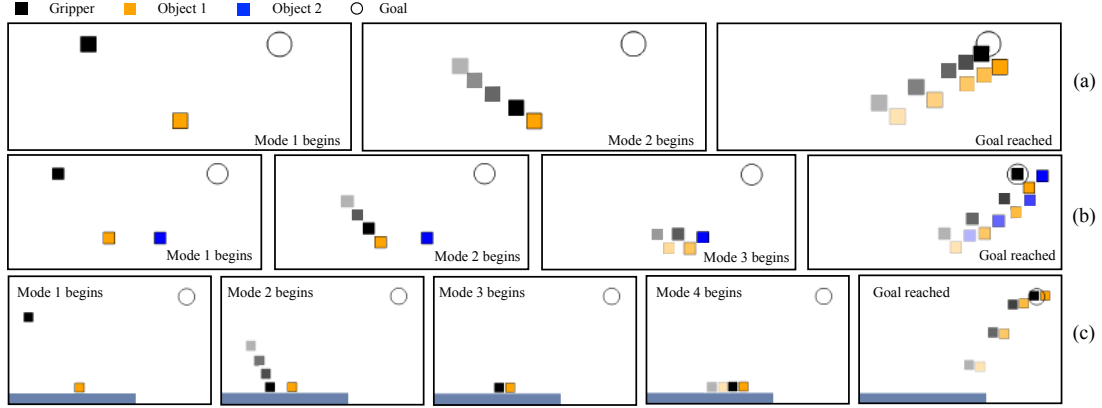


Figure 2.2: Learned behaviors for (a) 1-object pickup task, (b) 2-object pickup task, and (c) pickup task requiring sliding along the table to align with the object before picking it up. A 2D gripper (black block) picks up objects (orange and blue blocks) and takes it to goal region (black circle). Motion of gripper and objects is depicted by overlaid transparent frames.

replan using observations from the environment. We evaluate a posterior over the mode estimate using the current observation as

$$p(\boldsymbol{\rho}_t | \mathbf{x}_t, \mathbf{u}_{t-1}, \mathbf{x}_{t-1}) = z * p(\boldsymbol{\rho}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\rho}_t) \quad (2.3)$$

where $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\rho}_t)$ gives the probability of seeing the observed transition given a certain mode $\boldsymbol{\rho}_t$, $p(\boldsymbol{\rho}_t | \mathbf{x}_t)$ is the predicted mode from the learned classifier, and z is the normalization constant. This approach to evaluating the mode using observed state transitions is similar to the eMOSAIC approach (20). The above posterior can also be calculated using measured contacts if the robot is equipped with force/contact sensing. Using this posterior estimate of the mode, we replan as given in Algorithm 4. Adding this framework to our approach, we enable our controller to be predictive of mode transitions (using the classifier) as well as reactive (using measurements to evaluate posterior) to unanticipated mode changes and impacts conditions making the controller robust to learned model inaccuracies.

2.4 Experimental Results

The proposed method was evaluated in simulation using the Box2D environment and in the real-world on the 7DOF Franka-Emika Panda robot arm.

To evaluate our controller, which is both reactive and predictive and takes into account discontinuities due to contact, we compare our results to 1) a purely reactive eMOSAIC controller (20), 2) a purely predictive version of our controller, i.e. without using the posterior over the mode to replan using MPC, and 3) a predictive and reactive controller that doesn't take into account discontinuities due to contact.

Table 2.1: RMSE of learned masses and learned goal configurations.

	Masses (kg)	Goal configurations (m)
Task 1	0.029	0.095
Task 2	0.034	0.088
Task 3	0.009	0.026

Experiments were designed to test the robustness of these controllers to process noise, model inaccuracies and variations in shape geometry and initial configurations not seen during training. We use two metrics to compare results: 1) success rate - defined by whether the robot is able to pick up the object(s) of interest and take them to a goal region in a specified time 2) cost to perform the task, defined as the LQR cost: $\sum_{t=0}^T ((\mathbf{x}_t - \mathbf{x}_{f_{\rho_t}})^\top \mathbf{Q}_{\rho_t} (\mathbf{x}_t - \mathbf{x}_{f_{\rho_t}}) + \mathbf{u}_t^\top \mathbf{R}_{\rho_t} \mathbf{u}_t)$. We perform three tasks in simulation: 1) pick up one object and deliver to goal location 2) pick up two objects and deliver to goal location 3) pick up task while interacting with the ground to align with the object.

Expert demonstrations are collected by sampling the initial configuration for the gripper and fixed initial configurations for the objects. We use ground truth dynamic properties of the objects and manually defined cost functions to collect expert demonstrations for each task in simulation. By hand-tuning the cost matrices, we can control how much the gripper slowed down before picking up each object. For each of the tasks, the classifier and dynamics model are learned first using system identification as described in Section 2.3.2. Then, the cost matrices and goal conditions are learned as described in Section 2.3.4. For the eMOSAIC controller, these learned cost matrices and goal conditions are used to design independent infinite-horizon LQR controllers, one for each mode. The root mean squared error (RMSE) between the learned and ground truth values for the masses and goal configurations for each mode are summarized in Table 2.1. Additional details concerning training and experiments are provided in Appendix A.

Task 1: 1-object pickup

We consider the task of picking up an object using a 2D gripper and taking it to a goal region, specified as the black circle shown in Fig. 2.2a. The classifier learns to predict two modes. Mode 1 is active when the gripper is away from the object and mode 2 is active when the relative position between the gripper and the object is smaller than a learned threshold value. For this task, we observe that the classifier learns to predict contact a little after the actual impact occurs signifying imperfectness in the learned model. The masses for the gripper and object are set as 1kg and 9kg respectively in simulation. We perform multiple trials with varying initial configurations and shape parameters. As shown in Fig. 2.4, our controller with and without impact dynamics ($\mathbf{C}_{\rho_t, \rho_{t-1}}$) and eMOSAIC generalize well to new scenarios. Our controller is robust to local inaccuracies and eMOSAIC is unaffected by them as it is purely reactive. Both of these methods are successful in taking the object to the goal



Figure 2.3: Sequence of images showing the Franka robot arm performing a dynamic pickup task using our learned reactive and predictive controller. The controller generalizes to previously unseen locations of the object and performs the task with 80% success rate.

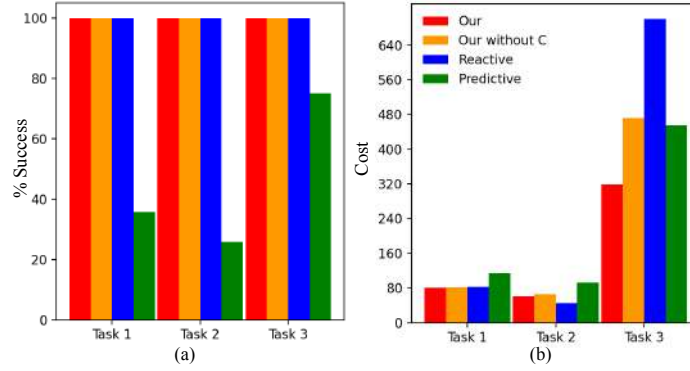


Figure 2.4: (a) Success rate (b) Execution cost for the 3 Box2D tasks

location in all the trials. On the other hand, performance of the predictive controller suffers because of inaccuracy in the learned classifier. The predictive controller fails to identify a contact even after the contact has actually occurred and does not switch to the next mode. We observe that the trajectories generated by eMOSAIC are non-smooth and the impact velocities are much higher than the expert’s, leading to higher cost. In contrast, the resulting cost of our controller is lower than the other controllers because of its capability to predict the next mode, curving the trajectory of the gripper in the direction of the next goal early on. The cost of our controller without $\mathbf{C}_{\rho_t, \rho_{t-1}}$ is also slightly higher as it doesn’t consider the discontinuous jump in velocities of the object and gripper upon contact.

Task 2: 2-object pickup

We consider the task of picking up two objects using a 2D gripper and moving them to a goal region, as shown in Fig. 2.2b. The model learns to identify three modes: Mode 1) relative position between the gripper and both objects is greater than a threshold value, Mode 2) relative position between the gripper and object 1 (orange) is smaller than a threshold value whereas between gripper and object 2 (blue) is larger than a threshold value, and Mode 3) relative position between the gripper and both

objects is smaller than a threshold value. The masses of the gripper and objects in simulation were set as 1kg each. For this task, we observe that the learned classifier predicts contacts a little earlier than when contact actually happens. As in Task 1, we observe that our controller with and without $\mathbf{C}_{\rho_t, \rho_{t-1}}$ is robust to these local inaccuracies, eMOSAIC is unaffected by it and both generalize well to new scenarios. As shown in Fig. 2.4 both these controllers were able to successfully perform the task in all trials. The success rate for the predictive controller suffers because in a large number of trials the gripper moves past the object without actually grasping it because the classifier wrongly predicted that grasp has happened. eMOSAIC performs better than our controller for this task because it is purely predictive. The inaccurate classifier leads our method to predict suboptimal trajectories in the short term, which leads to higher cost.

Task 3: Pickup task with ground interactions

This experiment evaluates the capability of our method to capture *intricate behaviors* from expert demonstrations. These demonstrations were designed to depict behavior that exploits contact to reduce uncertainty. We consider the task of picking up an object placed on a table. Fig. 2.2c demonstrates how the gripper first comes in contact with the table and moves along it before coming in contact with the object and pushing it along the table before picking it up. As before, we perform experiments by varying initial configurations and block sizes. We observe that the success rate for the predictive controller improves as compared to other tasks because alignment with the object before picking it up reduces uncertainty in relative positions. Our controller with and without $\mathbf{C}_{\rho_t, \rho_{t-1}}$ and eMOSAIC are successful in completing the task in every trial but the cost increases significantly if $\mathbf{C}_{\rho_t, \rho_{t-1}}$ is not considered. This is because the controller is unable to predict that the velocity of the gripper in the direction of contact goes to zero upon contact with the ground and the subsequent motion is along the ground. This leads to suboptimal predictions and thus higher cost. The cost for eMOSAIC is also high in this task. This is because the gripper and object overshoot the table due to the lack of a predictive component preparing the gripper to move up towards the goal early on, requiring a large force to correct the motion to eventually move toward the goal.

2.4.1 Robot Experiments

We consider the task of dynamically picking up an object and taking it to a goal location as shown in Fig. 2.3. Ten expert demonstrations were collected using human-guided kinesthetic teaching (35) with fixed object and goal locations. In the human demonstration, the robot slows down before contact, without stopping, to prevent the object from toppling. The location of the object was tracked using April tags and a Kinect depth sensor (36). The robot dynamics were linearized using Cartesian space feedback linearization. Due to a lack of reliable contact sensing, we only rely

on sensing of the object location as feedback to compare the performance of our controller with the reactive controller.

Using the method described in Section 2.3.2, the classifier learns to discretize the task into two modes. The goal condition for the first mode (coming in contact with object) was learned with RMSE of 0.015m, and the goal condition for the second mode (final configuration) was learned with a RMSE of 0.079m. To compare our controller to the reactive eMOSAIC controller we perform 10 trials with varying initial locations of object. Using our controller, the robot successfully picks up the object and takes it to the goal location 8/10 times. Average cost over successful trials with our controller is 1.1789. Using the purely reactive eMOSAIC controller, the robot successfully picks up the object and takes it to goal location 6/10 times. The average cost over the successful trials is 1.272. The sharp movements of the reactive controller right after coming in contact with the object is the main reason behind the failed attempts. Unlike in Box2D experiments, where abrupt motion in any direction can occur, such behavior does not perform well in real-world scenarios and can lead to failed grasps. As compared to eMOSAIC, our controller executes smooth trajectories before and after impact as it operates over all modes.

2.5 Conclusion

We presented a framework for developing a robust differentiable feedback controller for learned switching linear dynamical systems that operates over multiple dynamical modes, predicts and reacts to discontinuous mode switches and generalizes to unseen scenarios. Expert demonstrations inform mode predictions, contact dynamics, linear dynamical system parameters, cost functions and goal conditions. Experimental results show a significant improvement over eMOSAIC, a purely reactive controller, in multi-object pick-up, sliding and grasping, and real-robot grasping. We find that replanning using MPC and posterior modes improves robustness to model inaccuracies. In future work we will explore planning over modes to generalize to tasks that might require different mode sequencing. We plan to extend the capabilities of our controller to non-linear dynamical systems by learning locally linear dynamical models.

2.6 Acknowledgments

We gratefully acknowledge support for this work from the National Science Foundation under Grant No. CMMI-1925130 and IIS-1956163. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

3

DYNAMIC INFERENCE ON GRAPHS USING STRUCTURED TRANSITION MODELS

Enabling robots to perform complex dynamic tasks such as picking up an object in one sweeping motion or pushing off a wall to quickly turn a corner is a challenging problem. The dynamic interactions implicit in these tasks are critical towards the successful execution of such tasks. Graph neural networks (GNNs) provide a principled way of learning the dynamics of interactive systems but can suffer from scaling issues as the number of interactions increases. Furthermore, the problem of using learned GNN-based models for optimal control is insufficiently explored. In this work, we present a method for efficiently learning the dynamics of interacting systems by simultaneously learning a *dynamic* graph structure and a *stable* and *locally linear* forward model of the system. The dynamic graph structure encodes evolving contact modes along a trajectory by making probabilistic predictions over the edges of the graph. Additionally, we introduce a temporal dependence in the learned graph structure which allows us to incorporate contact measurement updates during execution thus enabling more accurate forward predictions. The learned stable and locally linear dynamics enable the use of optimal control algorithms such as iLQR for long-horizon planning and control for complex interactive tasks. Through experiments in simulation and in the real world, we evaluate the performance of our method by using the learned interaction dynamics for control and demonstrate generalization to more objects and interactions not seen during training. We introduce a control scheme that takes advantage of contact measurement updates and hence is robust to prediction inaccuracies during execution.

3.1 Introduction

Common everyday tasks such as picking up an object in one smooth motion, pushing a heavy door using the momentum of our bodies, or pushing off a wall to quickly turn a corner involve complex dynamic interactions between the human

and the environment. These dynamic interactions are critical in successful execution of these tasks. Thus, to enable robots to perform such dynamic tasks effectively, we need to consider the dynamics of the robot, the individual objects, as well as the interactions between them. One way to accomplish this is to explicitly learn these complex dynamics from data, and then use the learned models to intentionally leverage contacts to perform interactive and dynamic tasks.

Learning the interaction dynamics of multi-body systems is a challenging problem especially as the number of interacting bodies in the scene increases, resulting in a combinatorial explosion in the size of the state space as well as in the number of interactions to reason about. In recent work, graph neural networks (GNNs) (37; 38) have been used to model and learn *object-centric* interaction dynamics of such systems. These networks provide a principled way of learning the dynamics of large interactive systems by incorporating a strong structural prior (*relational inductive bias*) in the learned dynamics. However, many GNN based approaches still suffer from scaling issues as the number of interactions increases (39). For real-world dynamic manipulation tasks which usually involve a small set of interactions occurring at a given time in the scene, learning time varying sparse graph structures can help mitigate some of these issues.

In this work, we develop a method for efficiently learning the dynamics of interacting systems by simultaneously learning a relational *dynamic* graph structure and a *stable locally linear* forward dynamic model of the system. The learned dynamic graph structure encodes information about dynamics and contact modes evolving along the task trajectory by adding and removing edges from the graph as contact is made and broken. This approach allows for strong generalization of the learned dynamics when more nodes are added to the graph. Starting by representing our system as a fully connected graph, we learn time-varying and state dependent *edge activations* using a *graph inference module*. An edge is inferred to be active when the nodes connected via the edge interact and influence the forward dynamics, thus making it critical for the edge activations and the forward model to be learned together. When a new object is added to the scene, the graph inference module can predict whether or not the new node influences the system dynamics and the dynamics of other nodes based on edge predictions. This allows the learned model to generalize more effectively to new objects and new interactions in the environment.

We use the spring-mass-damper model as a structural prior over the *local* object-centric dynamics of our system. This provides us with two benefits: linearity and stability. For positive values of mass, stiffness and damping parameters, the linear spring-mass-damper model is always stable about the equilibrium point (which is also learned) (40). Assuming a locally linear and stable prior structure over our dynamics model allows us to represent a wide range of non-linear systems and use the learned dynamics for long-horizon planning and control by precluding the rollout trajectories from growing unbounded. The learned locally linear dynamics also enable the use of optimal control algorithms, such as iLQR, to exploit the interaction dynamics

and perform complex tasks optimally by predicting future edge activations and thus contact modes as well.

Control strategies that use a learned model are naturally sensitive to model inaccuracies, especially near the contact regions. This can lead to catastrophic failures during execution. To alleviate this issue, we use gated recurrent units (GRUs) to learn a probabilistic model that predicts edge activations at each time step. These units introduce a temporal dependency of the current edge activation on the previous edge activations. During execution, observed contact modes are used to evaluate a posterior over the edge activations and update them, which enables more accurate future rollouts. We use model predictive control (MPC) to replan using the observed current state and contact modes.

We also explore an exciting application of this work in the field of apprenticeship learning wherein the learned locally linear model can be used for learning the quadratic cost function underlying expert demonstrations using differentiable LQR as the policy class (1; 17). The learned behavior can then be generalized to unseen goal conditions.

The key contributions of this work are two-fold: 1) a method for learning stable locally linear dynamics for non-linear interactive systems using graph neural networks by encoding changing dynamics and contacts as part of the graph structure, enabling strong generalization properties to more objects in the scene, and 2) using the learned locally linear dynamics to devise a robust control scheme that utilizes the recurrent nature of the learned graph structure to adapt the model predictions and the policy to observed contact events.

We evaluate our results in simulation on multiple object dynamic pickup and dynamic door-opening tasks. We also perform real world experiments using the 7DOF Franka-Emika Panda robot arm for a dynamic pickup task ¹.

3.2 Related Work

Learning the interaction dynamics of contact-rich tasks is a problem of interest in many areas of robotics research such as manipulation (21), collaborative robotics and assistive systems (41). In some recent works the dynamics of such interactive systems is learned implicitly (42; 43), while in others (1; 11; 15; 29) the focus is on modelling these changing dynamics as explicit dynamic modes. In our method, we aim to exploit the inherent structure underlying interactions in physical systems by modelling them as graphs and taking advantage of the generalization benefits of graph-based approaches.

There has been tremendous progress towards the development of graph neural networks for modelling and learning the forward interaction dynamics of physical systems. Approaches such as (37; 38; 44; 45; 46) learn the dynamics model assum-

¹Supplementary video: <https://youtu.be/A9YG8VrIpgQ>

ing a static fully connected or known graph structure. Such models can require a large amount of interaction data and training time to accurately learn the interaction dynamics. There has also been work towards using static graphs for planning and control (47; 48; 49). We focus on the problem of simultaneously learning the dynamic graph structure and the forward dynamic model of the system in a purely unsupervised fashion and using that model for control.

Some approaches (50; 51; 52) learn a graph structure that is static over the entire task, or along a single task trajectory, while other methods learn to actively predict edge interactions (39; 53; 54) using attention mechanisms. We take inspiration from these approaches for learning probabilistic predictions of edge activations that evolve with the state of the system.

Learning to control non-linear systems using locally optimal control algorithms, such as iLQR, is an exciting area of research. Such methods require learning locally linear and stable dynamics models of the system. Some recent approaches that aim to learn locally linear dynamics of the system for control include (55; 56; 57; 58) but these approaches have rarely been extended to graph based interactive dynamic systems. We derive inspiration from these works and model our system dynamics as locally linear, which enables the use of optimal control algorithms such as iLQR. In our work, we also explore methods developed in (1; 17; 59) for utilizing the learned linearized dynamics for learning simple quadratic cost functions of differentiable controllers such as LQR.

3.3 Method

In this section, we discuss in detail our method for learning stable locally linear dynamics for non-linear interactive systems using graph neural networks. Our training pipeline is composed of two main modules that are trained together, 1) a graph inference module that learns the dynamic graph structure, and 2) a forward dynamics module that learns the stable locally linear forward dynamics given the graph structure.

3.3.1 Graph Inference Module

This module predicts the dynamic interaction graph structure — which edges in the dynamic graph are *active* or *inactive* at a certain time step. In our domain, where physical objects are represented as nodes in the graph, this module infers whether or not, at a certain time step, objects influence each other’s dynamics. For example, in a scenario where a robot gripper picks up an object, an edge connects the gripper and the object only after the object has been grasped.

The state of the system at time t is represented as a graph $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{E}_t)$ where nodes \mathcal{N}_t represent objects in the scene and edges \mathcal{E}_t connect interacting nodes. Node

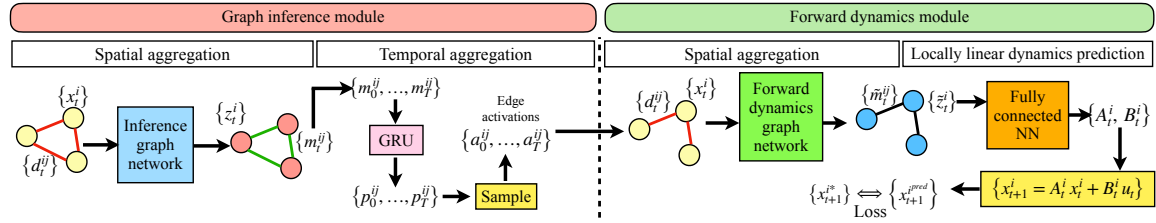


Figure 3.1: Method overview: Our method has two main parts 1) Graph inference module: we start with a set of fully connected graphs and pass them through the inference graph network that performs message passing between the nodes and outputs embedded graphs. Next, to aggregate information temporally, we pass the edge embeddings $\mathbf{m}_{0,\dots,T}^{ij}$ through a GRU that outputs a discrete probability distribution \mathbf{p}_t^{ij} over the edge types (active or inactive) for each edge. We sample from this distribution to get the edge activations \mathbf{a}_t^{ij} (see 3.3.1). 2) Forward dynamics module: using the edge activations, we remove the inactive edges from the fully connected input graphs and then pass them through the forward dynamics graph network that outputs another set of embedded graphs. The node embeddings $\tilde{\mathbf{z}}_t^{ij}$ are passed through a fully connected neural network that outputs stable locally linear transition dynamics (see 3.3.2). We forward propagate the learned dynamics using the input control and get the next state which is used for loss calculation during training (3.4).

$n_t^i \in \mathcal{N}_t$ is represented using a feature vector \mathbf{x}_t^i where $\mathbf{x}_t^i = [\mathbf{q}_t^i, \dot{\mathbf{q}}_t^i]$. \mathbf{q}_t^i and $\dot{\mathbf{q}}_t^i$ are the position and velocity of the i th object respectively. The edge $e_t^{ij} \in \mathcal{E}_t$ connecting nodes i and j is represented with edge features \mathbf{d}_t^{ij} where \mathbf{d}_t^{ij} is the distance between nodes n_t^i and n_t^j .

An overview of the graph inference module is shown in Fig. 3.1. We start with a set of trajectories of fully connected graphs $(\mathcal{G}_0, \mathbf{u}_0, \dots, \mathcal{G}_T, \mathbf{u}_T)$ where \mathbf{u}_t is the control applied at time t , and T is the length of the trajectory. In order to learn the dynamic graph structure, we first use a graph neural network to propagate information spatially over each of the graphs $(\mathcal{G}_0, \dots, \mathcal{G}_T)$ via message passing between the nodes using a node feature network f_n^{inf} and an edge feature network f_e^{inf} :

$$\begin{aligned}\mathbf{m}_t^{ij} &= f_e^{\text{inf}}(\mathbf{x}_t^i, \mathbf{x}_t^j, \mathbf{d}_t^{ij}) \\ \mathbf{z}_t^i &= f_n^{\text{inf}}(\mathbf{x}_t^i, \sum_j \mathbf{m}_t^{ij})\end{aligned}$$

where f_e^{inf} and f_n^{inf} are fully connected neural networks representing the *Inference GNN* and \mathbf{z}_t^i and \mathbf{m}_t^{ij} are the output node and edge embeddings respectively. After two steps of message passing, the output edge embeddings $\mathbf{m}_{0:T}^{ij}$ for each edge are passed through a gated recurrent unit (GRU) to allow temporal information flow:

$$\begin{aligned}\mathbf{p}_{0:T}^{ij} &= \text{GRU}_\phi(\mathbf{m}_{0:T}^{ij}) \\ \mathbf{a}_{0:T}^{ij} &= \text{sample}(\mathbf{p}_{0:T}^{ij})\end{aligned}\tag{3.1}$$

where ϕ are the trainable parameters of the GRU and \mathbf{p}_t^{ij} is a discrete probability distribution over the edge types. We consider two edge types: active and inactive. Sampling from probability distribution \mathbf{p}_t^{ij} gives us a one hot vector representing the *edge activation* \mathbf{a}_t^{ij} . If an edge is active at time t , $\mathbf{a}_t^{ij} = [1, 0]$ else $\mathbf{a}_t^{ij} = [0, 1]$. Naturally, this representation can be extended to include more than two edge types. Differentiability of this sampling procedure can be ensured by employing techniques such as Gumbel softmax (60). If an edge e_t^{ij} is sampled to be inactive it is removed from the graph \mathcal{G}_t for the subsequent calculations and no messages pass through that edge. The output of the graph inference module is a trajectory of graphs $(\tilde{\mathcal{G}}_0, \dots, \tilde{\mathcal{G}}_T)$ where $\tilde{\mathcal{G}}_t = (\mathcal{N}_t, \tilde{\mathcal{E}}_t)$ and $\tilde{\mathcal{E}}_t \subseteq \mathcal{E}_t$ is the set of active edges. Given this inferred graph structure, our next step is to model the forward dynamics of the system.

3.3.2 Forward Dynamics Module

An overview of the forward dynamics module is shown in Fig. 3.1. To learn the object-centric interaction dynamics of the system, we use another GNN to perform two steps of message passing over each of the graphs $(\tilde{\mathcal{G}}_0, \dots, \tilde{\mathcal{G}}_T)$ to propagate information spatially between connected nodes.

$$\begin{aligned}\tilde{\mathbf{m}}_t^{ij} &= f_e^{\text{dyn}}(\mathbf{x}_t^i, \mathbf{x}_t^j, \mathbf{d}_t^{ij}) \\ \tilde{\mathbf{z}}_t^i &= f_n^{\text{dyn}}(\mathbf{x}_t^i, \sum_j \tilde{\mathbf{m}}_t^{ij})\end{aligned}$$

where f_e^{dyn} and f_n^{dyn} are fully connected neural networks representing the *Forward Dynamics GNN*. Here, for scenarios where more than two edge types are considered, for example to capture varied behavior over active edges, one can use a different edge feature network f_e^{dyn} for each edge type. The output node embeddings $\tilde{\mathbf{z}}_t^i$ now encode interaction information between the connected nodes and can be used to learn the forward dynamic parameters of the interactive system independently for each node. To learn stable locally linear dynamics, we approximate the local dynamics of each node to resemble a spring-mass-damper system that is decoupled in its degrees of freedom. The parameters for each degree of freedom are derived from the node embeddings at each time step. This representation was chosen since the dynamics of a spring-mass-damper system can be constrained to be stable (bounded output) by constraining the mass, stiffness and damping parameters to be greater than zero (40). This is easily achieved by using activations such as sigmoid and relu. We use a fully connected neural network g^{dyn} to output the dynamic parameters for each node as,

$$\boldsymbol{\alpha}_t^i = g^{\text{dyn}}(\tilde{\mathbf{z}}_t^i) \quad (3.2)$$

where $\boldsymbol{\alpha}_t^i = [\frac{1}{m_t^i}, k_t^i, c_t^i, \hat{x}_t^i]$ where m_t^i is the mass, k_t^i is the stiffness of the spring, c_t^i is the damping, \hat{x}_t^i is the equilibrium point for node i at time t . These parameters can then be used to compute the stable locally linear dynamics of the system as:

$$\mathbf{x}_{t+1}^i = \mathbf{A}_t^i \mathbf{x}_t^i + \mathbf{B}_t^i \mathbf{u}_t + \mathbf{o}_t^i \quad (3.3)$$

where

$$\mathbf{A}_t^i = \begin{bmatrix} 1 & dt \\ -\frac{k_t^i dt}{m_t^i} & 1 - \frac{c_t^i dt}{m_t^i} \end{bmatrix}, \mathbf{B}_t^i = \begin{bmatrix} 0 \\ \frac{dt}{m_t^i} \end{bmatrix}, \mathbf{o}_t^i = \begin{bmatrix} 0 \\ \frac{k_t^i \hat{x}_t^i dt}{m_t^i} \end{bmatrix}$$

for a discrete-time one degree of freedom system.

3.3.3 Training

The graph inference module and the forward dynamics module are trained together to simultaneously learn the dynamic graph structure and the locally linear forward dynamics. We collect a set of N trajectories containing multiple interacting objects and represent them as graphs $\{(\mathcal{G}_0^*, \mathbf{u}_0, \dots, \mathcal{G}_{T-1}^*, \mathbf{u}_{T-1})_n\}_{n=0, \dots, N}$. The loss function for training the model is written as,

$$\mathcal{L}_{\text{dyn}} = \sum_i \|\mathbf{x}_{1:T}^{i*} - \mathbf{x}_{1:T}^{i\text{pred}}\|_2 + \sum_{i,j} \text{KL}[\mathbf{p}_{0:T-1}^{ij} \parallel \mathbf{q}_{0:T-1}^{ij}] \quad (3.4)$$

where \mathbf{x}_{t+1}^{i*} is the observed next state and $\mathbf{x}_{t+1}^{i\text{pred}} = \mathbf{A}_t^i \mathbf{x}_t^{i*} + \mathbf{B}_t^i \mathbf{u}_t + \mathbf{o}_t^i$ is the predicted next state for node n^i . $\mathbf{p}_{0:T-1}^{ij}$ is the probability distribution over predicted edge activations and $\mathbf{q}_{0:T-1}^{ij}$ is a prior on the edge activations. In prior work (50; 51), where the focus is on learning a static graph structure, the prior is designed such that

a sparse graph is learned. However, in our scenario, since the graph structure is not static and can be sparse or dense depending upon the state, we design a prior using the relative distances between the nodes. In particular, when two nodes are spatially close to each other, the prior suggests that the probability of the edge connecting these nodes being active, is high. In particular, for a given distance threshold of d_{th} , the prior over edge c_t^{ij} at time t is given by:

$$\mathbf{q}_t^{ij} = \text{Softmax}\left(\begin{bmatrix} d_{th} - d_t^{ij} \\ d_t^{ij} - d_{th} \end{bmatrix}\right)$$

where the edge feature d_t^{ij} is the distance between nodes n_t^i and n_t^j . The neural networks f_e^{inf} , f_n^{inf} , GRU_ϕ , f_e^{dyn} , f_n^{dyn} and g^{dyn} are trained simultaneously using stochastic gradient descent to minimize the loss \mathcal{L}_{dyn} .

3.3.4 Execution

While it is common practice to use model predictive control for replanning using the current observed state, for interactive tasks where contacts are critical, it is essential to utilize observed contact information for replanning as well. Since we use GRU to learn the edge activations, it introduces a temporal dependency of the current edge activation on the previous edge activations. For one time step prediction, this recurrent process looks as follows, $\mathbf{p}_{t+1}^{ij} = \text{GRUCell}(\mathbf{p}_t^{ij}, \mathbf{m}_t^{ij})$. During execution, using the observed contact modes, we update the current probabilistic edge activations by calculating a posterior on the edge activations as follows,

$$\tilde{\mathbf{p}}_t^{ij} = z \mathbf{p}_t^{ij} \mathbf{c}_t^{ij}$$

where \mathbf{c}_t^{ij} is a discrete probability distribution representing the observed contact between node n_i and n_j at time t and z is the normalization constant. Since GRU aggregates past information for future rollouts, the updated edge activation $\tilde{\mathbf{p}}_t^{ij}$ allows for more accurate future predictions. In particular, we use the posterior edge activations $\tilde{\mathbf{p}}_t^{ij}$ to make updated future predictions as follows, $\mathbf{p}_{t+1}^{ij} = \text{GRUCell}(\tilde{\mathbf{p}}_t^{ij}, \mathbf{m}_t^{ij})$.

3.4 Experimental Results

The proposed method is evaluated in simulation using Box2D and Isaac Gym environments and in the real-world on the 7DOF Franka-Emika Panda robot arm. Through our experiments, we aim to evaluate 1) How useful is our graph inference module in enabling efficient and accurate learning of the dynamics of the interactive system? 2) What are the generalization properties of the learned dynamics model as more objects (not seen during training) are added to the system? 3) How robust

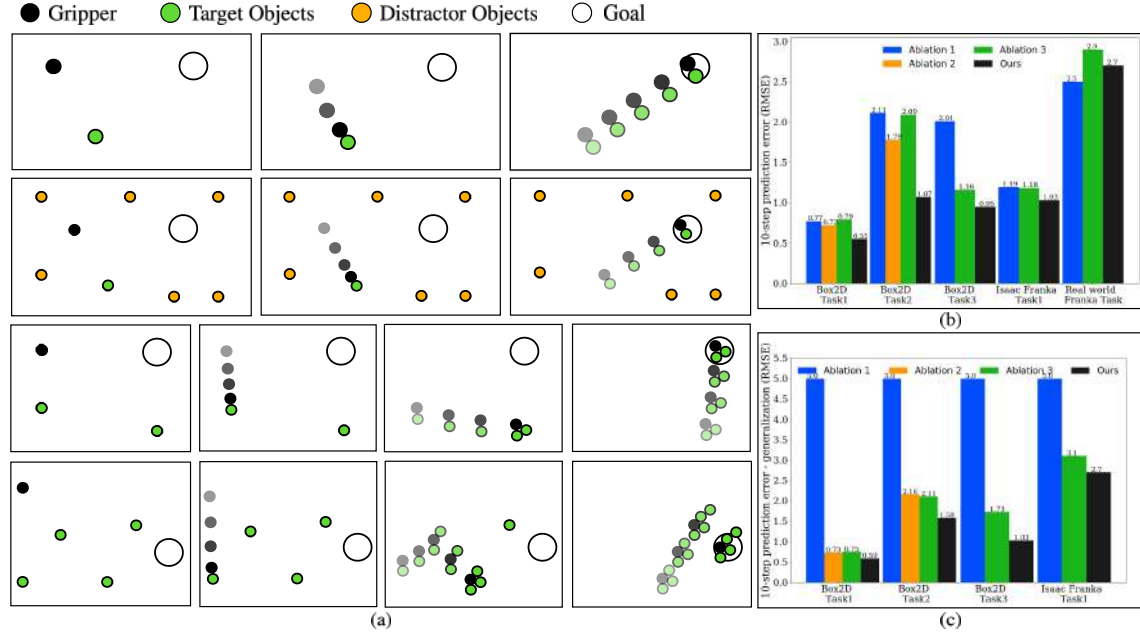


Figure 3.2: (a) Row 1: Box2D Task 1 with same number of objects as seen during training. Row 2: Box2D Task 1 generalization scenario with many distractor objects (orange) in the scene. Row 3: Box2D Task 2 with same number of objects as seen during training. Row 4: Box2D Task 2 generalization scenario for picking up many more objects than during training. (b) N-step root mean square error between the predicted and executed trajectories. These experiments are performed with the same number of objects and interactions as seen during training. (c) N-step root mean square error between the predicted and executed trajectories. These experiments are performed to test for generalization by including many more objects and interactions in the task than seen during training.

is our execution strategy to inaccuracies in the learned model? 4) Can our learned stable linearized dynamics be used for learning the cost function of a differentiable LQR controller using expert demonstrations of a task?

Ablation studies: To answer the first three questions we perform the following three ablation studies: 1) Training the dynamic model without the graph inference module i.e. learning the forward dynamics module given fully connected graphs. The dataset for this study is the same as the dataset used for training the full model. We call this model **No-GIM**. 2) Training the dynamic model without the graph inference module but with an augmented dataset containing more objects and interactions. We call this model **No-GIM-Aug**. **No-GIM** and **No-GIM-Aug** are similar to prior work (37; 44; 47) that assume a static fully connected or known graph structure. 3) Training the full dynamic model with the graph inference module but replacing the GRU in (3.1) with a fully connected neural network that takes as input only the

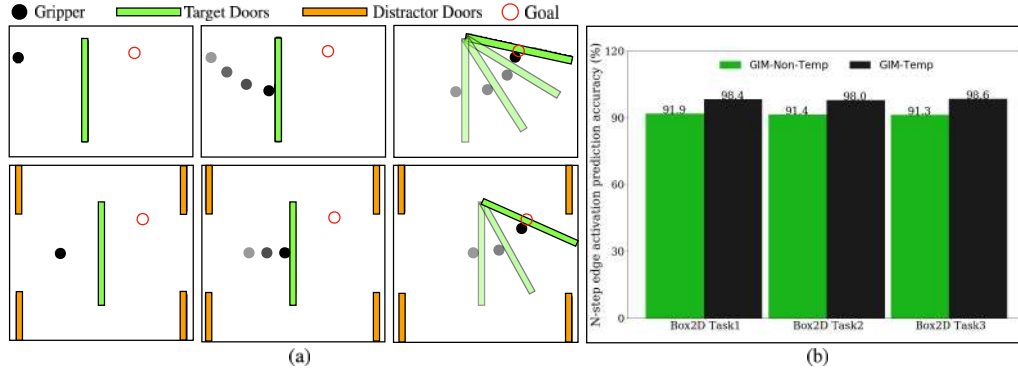


Figure 3.3: (a) Row 1: Box2D Task 3 with same number of doors as seen during training. Row 2: Box2D Task 3 generalization scenario with many distractor doors in the scene. (b) N-step edge activation prediction accuracy.

current edge embedding \mathbf{m}_t^{ij} to output the edge activation i.e. there is no temporal relationship between the predicted edge activations. We call this model **GIM-Non-Temp**. Due to this modification in the network architecture, we will no longer be able to utilize the robust control scheme elaborated on in 3.3.4 since posterior on the edge activations can no longer be used to update future predictions. We call our model **GIM-Temp**. In addition to the above ablation studies, we also train a model **Full-AB** that uses our graph inference module but instead of using a spring-mass-damper system based stable dynamics model in the forward dynamics module, learns the full **A** and **B** matrices from scratch.

Application: Apprenticeship learning. To answer question 4, given the learned stable locally linear interaction dynamics, we learn the cost matrices **Q** and **R** of a discrete-time linear quadratic regulator (LQR) using expert demonstrations. We demonstrate generalization of learned expert behavior to goal regions not visited by the expert. We closely follow the cost learning method in (1), but instead of learning a different cost function and goal condition for each dynamic *mode*, modeled as a discrete global variable, we learn a single cost function for a certain task with a fixed goal. **Evaluation metrics:** To test for generalization, all the ablation models mentioned above, including our model, are tested with more objects and interactions than seen during training. For evaluating the accuracy of the learned model, the generalization properties and the effectiveness of the robust control scheme, we use N-step prediction error as the metric. This error is calculated upon using the learned locally linear models for control using iLQR-MPC (receding horizon control) and calculating the root mean square error between the predicted trajectory and the executed trajectory for a rollout length of N . For each environment, N is calculated such that the rollout time is 0.5 seconds i.e $N = 0.5/dt$. The prediction error is calculated over all the nodes in the graph. For **GIM-Non-Temp**, in addition to N-step prediction error, we also evaluate the accuracy of the predicted edge activations for a

rollout length of N and compare it with our method **GIM-Temp**. We get the ground truth edge activations by observing contacts during execution. Contact information is not used during training. For **No-GIM-Aug**, we also present results for the sample complexity, i.e how much more interaction data is needed in order to achieve similar level of performance as with our method. This will demonstrate the efficiency of our method as compared to models where the graph inference module is not used.

Fig. 3.2(b) shows the N -step prediction error when the learned models are tested in environments with the same number of objects and interactions as seen during training. Fig. 3.2(c) shows the N -step prediction error for generalization scenarios i.e. testing with more interactions and more distractor objects in the scene. Fig. 3.3(b) shows the N -step edge activation prediction accuracy.

Box2D Tasks: We consider three tasks in this environment: **Task 1:** Dynamically (without stopping) picking up a 2D object using a 2D gripper and taking it to a goal region (Fig. 3.2(a) Rows 1 & 2). This task aims to assess if the learned graph inference module generalizes well to non-interacting objects in the scene (no edges to non-interacting objects) by testing with up to ten distractor objects. For this task, **No-GIM**, **GIM-Non-Temp** and **GIM-Temp** are trained on a dataset consisting of 40 trajectories of the gripper picking up *one* target objects and taking it to goal location. Initial positions of the gripper and object and goal locations are sampled to collect data. **No-GIM-Aug** is trained with an augmented dataset consisting of up to three distractor objects which do not interact (160 trajectories). **Task 2:** Dynamic pickup of multiple objects (Fig. 3.2(a) Rows 3 & 4). This task is designed to test for generalization to more *interactions* than seen during training. We test this by picking up upto five objects during testing while only two objects were picked up in the training dataset. This generalization is non-trivial, as compared to task 1 where distractor objects are ignored, since the model needs to have learned the dynamics behind the interactions that occur when an object is picked up and then generalize it to additional such interactions in the trajectory. For this task, **No-GIM**, **GIM-Non-Temp** and **GIM-Temp** are trained on a dataset consisting of 40 trajectories of the gripper picking up *two* target objects and taking them to a goal. **No-GIM-Aug** is trained with an augmented dataset consisting of trajectories where the gripper picks up a maximum of three objects and takes them to a goal (120 trajectories). **Task 3:** Dynamic door opening (Fig. 3.3(a)). This task aims to access the model’s ability to learn locally linear dynamics of a highly non-linear system and use the learned dynamics for control. We add upto ten distractor objects in the test setup of this task as well. For this task, **No-GIM**, **GIM-Non-Temp** and **GIM-Temp** are trained on a dataset consisting of 40 trajectories of the gripper dynamically opening a door.

Isaac Franka Task: We consider the task of dynamically picking up an object of mass 0.5kg using a 7DOF Franka-Emika Panda arm and taking it to a goal location (Fig. 3.4(a) and (b)). The robot is controlled using task space impedance control (35) during data collection wherein the orientation of the gripper is kept fixed. The initial cartesian position of the gripper, object (on a table) and goal are sampled for data

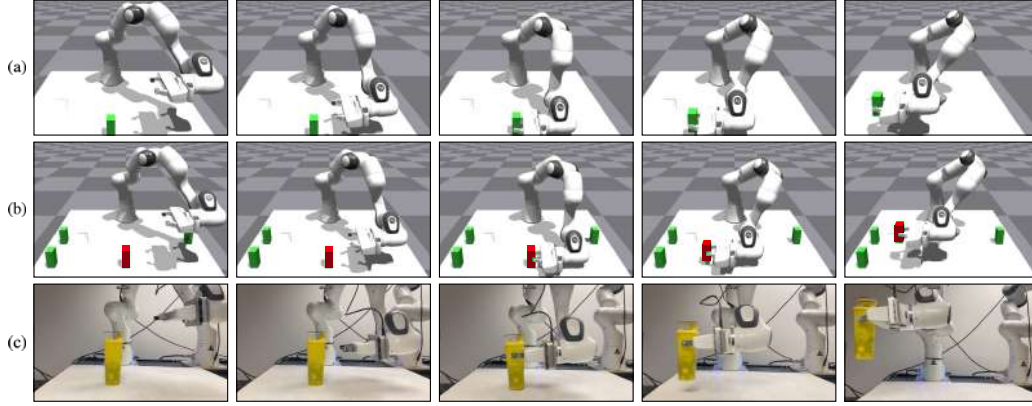


Figure 3.4: (a) Isaac Gym Task with same number of objects as seen during training. (b) Isaac Gym Task generalization scenario with distractor (green) objects in the scene. (c) Real world experiments with the 7DOF Franka-Emika Panda arm performing a dynamic pickup task using our learned model.

collection. **No-GIM**, **GIM-Non-Temp** and **GIM-Temp** are trained on a dataset consisting of 200 trajectories of the gripper picking up the target object and taking it to a goal. A cartesian space model of the robot is learned while assuming fixed orientation. We test for generalization in environments with up to five distractor objects.

Real robot experiments: Real world experiments are performed for the task of dynamically picking up an object of mass 0.83kg and taking it to a goal location using a 7DOF Franka-Emika Panda robot arm (Fig. 3.4(c)). The setup, controller, data collection and training strategy is the same as for the Isaac Gym Franka pickup task. The position and velocity of the object were tracked using April tags and a Kinect depth sensor (36). Due to lack of reliable contact sensing, we do not update edge activation predictions using observed contacts during execution but still rely solely on GRU for edge activation predictions. **No-GIM**, **GIM-Non-Temp** and **GIM-Temp** are trained on a dataset of 25 trajectories.

Training parameters and network architecture: All the models are trained using ADAM (61) with default parameters and a learning rate of $1e - 5$. The inference GNN and forwards dynamics GNN are composed of two fully connected neural networks each having two layers with 64 units each and ReLU activation. g^{dyn} in (3.3) is also a fully connected neural network having two layers with 64 units each and ReLU activation. We use *GRUCell* in PyTorch (62) and apply a softmax to its output for learning the edge activation probabilistic model.

3.5 Conclusion

In this work, we demonstrated the benefits of learning the forward model of interactive systems by simultaneously learning a dynamic interaction graph and a stable locally linear forward dynamic model given the graph. We showed that using a spring-mass-damper system as an approximation to the local dynamics allows us to effectively learn the parameters of a stable system that can be used for long-horizon planning. We then elaborated on the generalization benefits of learning a dynamic graph structure over using a fully connected graph and demonstrated that by introducing a temporal dependency of the current edge activation on the previous edge activations, we allow for contact measurement updates, ensuring more accurate future predictions. As an application of our work, we also discussed how our learned model can be used to learn the quadratic cost function underlying expert demonstrations using the method described in (1; 17). The learned behavior can then be generalized to unseen goals.

3.6 Acknowledgments

We gratefully acknowledge support for this work from the National Science Foundation under Grant No. CMMI-1925130 and IIS-1956163. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

4

MULTI-RESOLUTION SENSING FOR REAL-TIME CONTROL WITH VISION-LANGUAGE MODELS

Leveraging sensing modalities across diverse spatial and temporal resolutions can improve performance of robotic manipulation tasks. Multi-spatial resolution sensing provides hierarchical information captured at different spatial scales and enables both coarse and precise motions. Simultaneously multi-temporal resolution sensing enables the agent to exhibit high reactivity and real-time control. In this work, we propose a framework, MResT (Multi-Resolution Transformer), for learning generalizable language-conditioned multi-task policies that utilize sensing at different spatial and temporal resolutions using networks of varying capacities to effectively perform real time control of precise and reactive tasks. We leverage off-the-shelf pretrained vision-language models to operate on low-frequency global features along with small non-pretrained models to adapt to high frequency local feedback. Through extensive experiments in 3 domains (coarse, precise and dynamic manipulation tasks), we show that ourT approach significantly improves ($2\times$ on average) over recent multi-task baselines. Further, our approach generalizes well to visual and geometric variations in target objects and to varying interaction forces.

4.1 Introduction

Performing robotic manipulation tasks in the real world often requires using sensing modalities at different *spatial resolutions*. For instance, for peg-insertion, the robot can use a statically-mounted third-person camera (low spatial resolution or global information) to reach close to the hole, use a wrist-mounted first-person camera for finer alignment, and finally use proprioception and force-feedback for insertion (high spatial resolution or local information). Additionally, each sensing modality can be utilized at a different *temporal resolution*. For example, for coarse quasi-static subtasks (“reach hole”), using third-person camera images at a *low frequency* can be sufficient. However, finer reactive subtasks (“insert peg”), might require *high-*

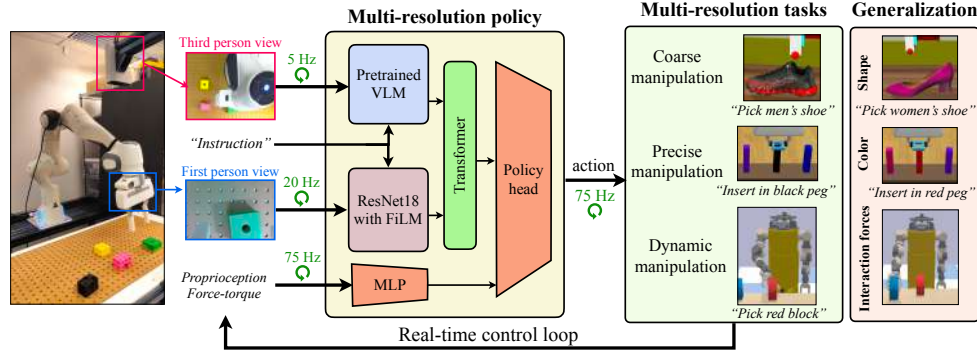


Figure 4.1: Our proposed approach uses sensing at different spatial and temporal resolutions for real time control of coarse, precise and dynamic tasks while enabling generalization to novel visual features and interactions.

frequency force-torque feedback. Based on this insight, we propose a multi-resolution (spatial and temporal resolution) sensor fusion approach for coarse quasi-static as well as precise reactive manipulation tasks.

Multi-resolution sensor fusion can enable generalization to novel visual-semantic targets. For instance, by utilizing global information from third-person camera images only for coarse localization and relying on local information from in-hand cameras and force-torque feedback for finer motions, the policy can learn to generalize to novel objects. Previous approaches to learning generalizable policies either require extensive data collection (63; 64; 65) or rely on pretrained models (66; 67; 68; 69) for policy adaptation (70). However, such approaches typically utilize a single sensory modality, while others that incorporate multiple sensors do not prioritize generalization (71). In our work, we avoid extensive data collection and instead leverage pretrained vision-language models in our multi-resolution approach to learning generalizable language-conditioned multi-task policies.

Although pretrained vision or vision-language models (VLMs) provide impressive generalization capabilities and enable learning language-conditioned multi-task policies, using large VLMs can have certain disadvantages. First, given their large size (e.g. Flamingo has 80B parameters (68)), they have slow inference which makes them unusable for real-time closed-loop control which is necessary for reactive tasks. Second, since pre-trained models are often trained on out-of-domain data, using them to solve in-domain manipulation tasks (especially precise tasks) may require finetuning (72). However, task-specific finetuning can make models less robust with reduced generalization (73).

To overcome the above challenges of utilizing large pretrained VLMs for real-time control of reactive tasks, we propose a framework that incorporates different capacity networks (that operate on different sensing modalities) at different frequencies. Specifically, we use large pretrained VLMs with slow inference at a lower frequency while small networks with fast inference at a higher frequency. Our low-frequency

pretrained VLMs operate on statically mounted third-person views and can provide global coarse feedback (such as approximate object locations) that is usually only needed at a low rate. On the other hand, we propose using small trained-from-scratch models with first-person camera views and force-torque data to obtain the high-frequency fine-grained feedback necessary to perform precise and reactive tasks. Further, to overcome the challenge of loss in generalization when finetuning pretrained VLMs, we *freeze* the pretrained VLMs to avoid losing their robustness and maintain their generalization abilities. Overall main contributions include:

- a framework for learning generalizable multi-task policies that incorporates multiple sensory modalities to capture global to local spatial information,
- combine sensor modalities at different frequencies to avoid bottlenecks and enable reactive control which we show empirically is essential for dynamic tasks,
- comprehensive experiments across 3 domains (and 2 real-world tasks) that include coarse, precise and dynamic manipulations tasks, and
- effective generalization across semantic task variations in both simulation and real-world.

4.2 Related Work

Vision-Language Pretrained Models for Robot Manipulation: Many prior works combine vision and language for robotic tasks. While early works focus on tabula-rasa learning (74; 75; 76), more recent works, use pretrained large language models (LLMs) and show efficient learning and improved generalization for robotics tasks (77; 78; 79; 80; 81). Many recent works also combine large general-purpose pretrained vision or vision-language models (VLMs) (66; 68; 82) for manipulation (70; 72; 83; 84; 85; 86; 87; 88; 89). Our work is more closely related to these latter works in that we also use pretrained VLMs for robot manipulation. Among these works, many works only use language for task-specification and do not focus on the generalization provided by pretrained models (88; 89). Additionally, other works adapt the pretrained representation for the downstream task (72; 86?). However, as we show empirically, such updates lead to representation drift and a loss of robustness for the pretrained general-purpose VLM. Hence, we propose not updating the pretrained representations. While (70; 87) use frozen VLMs, (87) only uses pretrained VLM as an open-world object detector to get pixel targets for the task at the first episode step. On the other hand, (70) uses the pretrained VLM with templated pick-and-place actions for manipulation. By contrast, we use VLMs in our multi-resolution framework with continuous feedback for reactive manipulation tasks.

Multi-Spatial Resolution for Robot Manipulation: Many prior works use multiple sensor modalities for robot manipulation, wherein each modality operates

at a different spatial resolution. For instance, prior works often combine visual (low spatial resolution) and proprioceptive (high spatial resolution) feedback (90; 91; 92), use wrist-mounted cameras for visual servoing (93; 94; 95) or for contact-rich manipulation tasks (96; 97; 98; 99), while other works focus on combining vision and haptic sensing (100; 101; 102; 103). Our work is similar to the first set of works i.e. we use both third person and first person cameras for precise manipulation. However, unlike most prior works (96; 99) which focus on single-task settings, we focus on multi-task settings and fuse multiple sensing modalities at different resolutions.

Multi-Temporal Resolution for Robot Manipulation: Learning reactive policies requires the robot to operate at high frequencies. Some recent works in robot manipulation focus on learning policies at different temporal resolutions. For instance, (104) decompose a manipulation task into different phases (e.g. visual reaching phase and tactile interaction phase) and learn separate policies for each phase as well as a blending policy. While (105) avoid the discrete formulation of an MDP and instead learn a continuous differential equation (106; 107) to model the low resolution features. By contrast, we use the discrete formulation and instead of decomposing policies into different phases we reuse features from low-resolution signals while operating at a high temporal resolution.

Dynamic Reactive Manipulation: Many prior works in robot manipulation focus on quasi-static tasks (63; 79). However, there has been increased interest in solving tasks that are reactive and dynamic in nature (1; 108; 109). Previous works focus on explicitly learning the dynamics (1) or using analytical models (108; 110) of such systems for achieving reactivity. These works often assume access to the ground truth object pose and are limited to a single-task setting. In our work, we learn how to perform such dynamic and reactive tasks using visual inputs in a multi-task setting.

4.3 Proposed Approach

In this section, we discuss our approach for learning a generalizable language-conditioned multi-resolution multi-task policy for precise and reactive manipulation tasks. Below, we provide details on how we utilize different sensing modalities and then delineate our training/inference and discuss how our approach enables real time control for reactive tasks while generalizing to novel tasks.

4.3.1 Multi-Resolution Architecture

Figure 5.2 shows the architecture of our multi-resolution approach. Our model takes as input multiple sensing modalities with different spatial resolutions, i.e., statically-mounted third-person camera view, first-person camera view and high frequency force-torque feedback. Each input is first processed separately before being

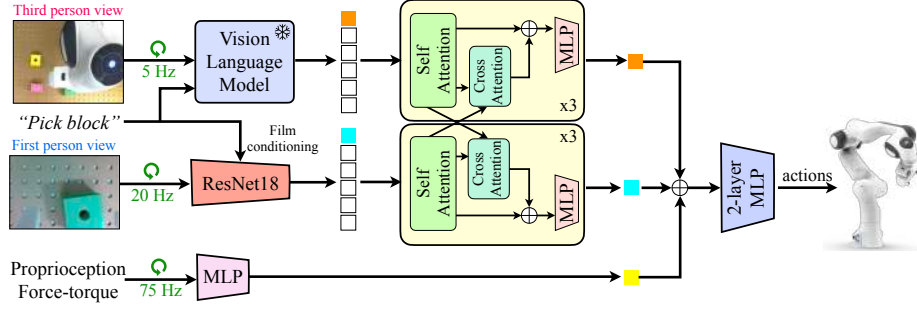


Figure 4.2: Overall architecture: Global low frequency information is extracted from third-person camera images using slow inference networks, local high frequency information is extracted from first-person camera images and proprioceptive, force-torque feedback using fast inference networks. These sensing modalities are then fused at different frequencies to enable real time high frequency control.

fused together at different temporal resolutions to output high frequency robot actions. Below we expand on each component of our architecture.

Low-Spatial Resolution Model: We use a low-spatial resolution sensor (third-person camera) to provide global task information to our agent. We use pretrained visual-language models to extract this global information from third-person views as well as to enable language-conditioning in a multi-task setting. Such pretrained models enable generalization to novel semantic features such as new objects or novel language commands. However, to ensure the pretrained model maintains its robustness we keep it *frozen*. However, using large VLMs to extract this generalizable global information comes with the drawback that the inference speed is very slow ($\approx 5\text{Hz}$). We experiment with two models CLIP (66) and MDETR (111) (language-conditioned DETR (112)), which use image-level and object-level information respectively.

High-Spatial Resolution Model: To ensure reactivity in the face of slow inference of pretrained VLMs, we use a smaller non-pretrained vision model (ResNet-18) (113) to process the first-person camera view at a higher frequency ($\approx 20\text{Hz}$). This view provides us with high-resolution local spatial information. To provide appropriate task-context to the first-person view we use small FiLM layers (114) for language conditioning. We train this model from scratch with augmentations (explained in the next paragraphs) to extract local spatial features that are useful for precise tasks. While using a small vision model enables faster processing it can still be insufficient for some highly dynamic tasks. Hence, we process the force-torque feedback and proprioceptive information at a much higher frequency ($\approx 75\text{Hz}$) using a small linear layer.

Multi-Resolution Sensor Fusion: We combine local and global sensing information (spatial resolutions) mentioned above at different temporal resolutions based on the capacities of the respective networks. Specifically, we reuse features (network activations) from lower frequency (third-person and first-person views) networks to match the frequency of the highest frequency (force-torque feedback) network. Doing

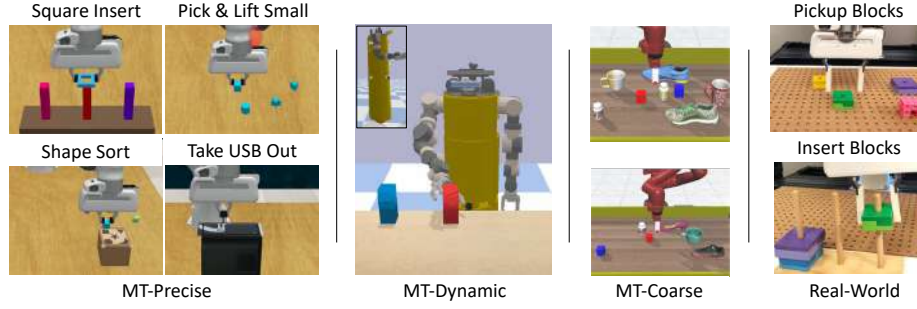


Figure 4.3: Task settings for evaluating our proposed approach. *Left*: Precision tasks. *Middle-left*: Dynamic tasks. *Middle-right*: Coarse tasks. *Right*: Real world pick and insertion tasks.

this ensures that the policy network outputs actions at a high frequency (equal to the frequency of the force-torque feedback network), thus enabling real-time control.

In addition to temporal-sensor fusion we also spatially fuse local and global sensing information, i.e. we fuse information extracted from third-person views with first-person view information and vice-versa. We achieve this using two small camera-specific transformers together with cross-attention. Each transformer uses self-attention within each modality (for its associated camera view) and cross-attention with the other modality (other camera view). As shown in Figure 5.2, we readout the CLS token from each transformer and concatenate them with the force-torque and proprioception embedding. This concatenated embedding is then processed using a 2-layer MLP policy head to output the robot actions. Please refer to Appendix C.2 for further details on the architecture.

Data Augmentations: Data augmentations have been shown to be helpful for single-task learning of manipulation tasks (99; 115). However, naively using image augmentations can be detrimental for learning generalizable multi-task policies. This is because pixel-level augmentations, such as color-jitter, grayscale etc., can result in semantic changes in the overall scene. Such semantic changes can lead to mismatch between the input image and language instruction provided for the given task. For instance, a demonstration shows “move to red block” but pixel augmentations can change the red block’s color. To avoid this while being able to utilize the benefits of augmentations we propose to use two different sets of augmentations. First, for third-person cameras we *only* use image-level augmentations (e.g. random crops, shifts). This avoids mismatch between image-and-text instructions and allows visual-language grounding from pretrained VLM to be utilized. Second, for first-person camera we use both image-level and pixel-level augmentations (color-jitter, grayscale). Since these augmentations lead to image-text mismatch this further enforces our agent to use the third-person camera view for coarse localization, while only relying on the in-hand view for finer precise motions. Using strong pixel-level augmentations on first-person view further make the in-hand model invariant to texture but rely more on edges and corners (116). This, as we show empirically, improves the generalization performance

of our model on heldout object variations.

Training and Inference: We use behavior cloning from expert demonstrations to train our model. We record data from each sensor at their respective frequencies. Specifically, camera images are recorded at 30 Hz and force-torque feedback at 250Hz. To match slower processing times of larger models during inference we sub-sample the third-person camera images to 5Hz and first-person camera images to 20Hz. We use AdamW (61) optimizer with learning rate $1 \times e^{-4}$ and weight decay 0.01. We train our model for 60 epochs, using a linear warmup, starting with learning rate 0, for 5 epochs and then decay the learning rate using a cosine-scheduler. We use a GTX-1080Ti for inference. Overall our architecture has $\approx 250M$ parameters. The pretrained vision-language model has $\approx 150M$ parameters (for MDETR) with an inference time of ≈ 0.1 seconds. The first-person camera model has $\approx 25M$ parameters with an inference time of 0.04 seconds. Finally, the force-torque and proprioception model along with the policy head have a total of $\approx 250K$ parameters with an inference time of ≈ 0.005 seconds. This allows the actions to be inferred at a max frequency of $\approx 200\text{Hz}$ although we use it at a reduced frequency of $\approx 75\text{Hz}$ which was sufficient for our tasks.

4.4 Experimental Setup

We first identify the key research questions that we aim to evaluate:

Q1: How does **multi-spatial resolution** sensing benefit learning language-conditioned multi-task (MT) manipulation policies for precise tasks? Specifically, we aim to evaluate the utility of multi-spatial resolution sensing for tasks that involve visual occlusions, partial observability, and precision.

Q2: How does **multi-temporal resolution** sensor fusion benefit learning reactive manipulation tasks? Specifically, we evaluate how our architecture enables closed loop control for reactive tasks.

Q3: How well does our approach **generalize** to tasks with novel visual-semantic targets? Specifically, we evaluate our approach’s robustness to distribution shifts, e.g., object colors and geometries.

4.4.1 Environments

To evaluate the above questions we use three task settings, 1) **MT-Precise:** Precise manipulation tasks, 2) **MT-Dynamic:** Dynamic manipulation tasks, and 3) **MT-Coarse:** Coarse table-top manipulation tasks. Below we detail each environment and discuss its usage to answer above questions.

MT-Precise For precise manipulation we use 4 spatial precision tasks from RL-Bench (117) (see Figure 4.3 (Left)) – square block insertion, pick up small objects, shape sorting, and unplug usb. We use this task domain to answer **Q1**. Specifically, we evaluate the need for multi-spatial resolution sensing in manipulation tasks that

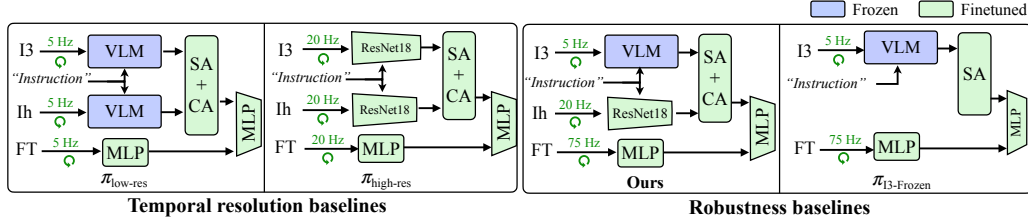


Figure 4.4: Temporal resolution and robustness baselines used to compare our multi-resolution approach.

require precise feedback and have partial observability, i.e., objects can go out of view of the first-person camera.

MT-Dynamic: We use the CMU ballbot (118) platform to perform dynamic pickup tasks in simulation (Figure 4.3 (Middle-Right)). We choose ballbot since it is a highly dynamic robot with an omnidirectional base (ball) capable of performing fast, reactive and interactive tasks. We consider the task of dynamically picking up an object, which requires quick reaction to contact with the object and grasping it to prevent toppling the object over. We use this setting to answer **Q2**.

MT-Coarse: We consider a canonical table-top manipulation setting ((119; 120)) involving coarse pick-and-place manipulation tasks with diverse objects – blocks, shoes, mugs, cylinders. We use this environment to answer **Q1** and **Q3**. Specifically, for **Q1** we contrast these coarse manipulation tasks with high precision tasks to evaluate the utility of multi-spatial resolution sensing.

Real-World Setup: We evaluate our approach on two real-world tasks. For precise manipulation (**Q1**) we use an *insertion* task to insert different blocks into cylindrical pegs (Figure 4.3 (Right top)). We also evaluate generalization abilities (**Q3**) using a *pickup* task, wherein we use 2 train objects and evaluate the learned policy on 8 objects with different geometry (shape, size) and visual (color, texture) features. Additional details on each environment are provided in Appendix C.1

4.4.2 Baselines

We compare our approach against recent methods which focus on learning generalizable policies in multi-task settings. We compare against RT-1(63) which proposes a transformer based policy and also against BC-Zero (64) which uses language conditioning using FiLM (114). However, both (63; 64) focus on coarse manipulation tasks and operate at a single-resolution (both temporal and spatial). To the best of our knowledge no prior work focuses on a multi-resolution approach for multi-task learning. Hence, to highlight the benefit of each component of our approach and answer the questions posed in Section 3.4 we modify our approach along different axes and propose additional baselines below.

Spatial Resolution baselines: To verify the utility of multiple spatial resolutions (**Q1**) we modify our approach and *remove* one sensory modality at a time. We use π_{-Ih} , π_{-I3} , π_{-FT} to refer to policies which *remove* first-person (**hand** view), **3rd**

person view and force-torque respectively.

Temporal Resolution baselines: To answer **Q2** we compare against single temporal-resolution approaches (Figure 4.4 (Left)), i.e., where all modalities (including force-torque) operate at the same frequency. We introduce two baselines, 1) $\pi_{\text{high-res}}$: small models with fast inference for both cameras (20Hz), and 2) $\pi_{\text{low-res}}$: larger models with slow inference for both cameras (5Hz).

Robustness baselines: We compare visual-semantic generalization ability of our approach (**Q3**) against two baselines (Figure 4.4 (Right)): 1) $\pi_{\text{multi-res-FT}}$: Finetune the pretrained VLM model, 2a) $\pi_{\text{I3-Frozen}}$: Uses only third-person camera (and force-torque) and keeps the pretrained model frozen. 2b) $\pi_{\text{I3-FT}}$: Uses only third-person camera (and force-torque) but finetunes the pretrained model.

Metrics: We use task success as the evaluation metric and report mean success over all tasks. During training, we evaluate the policy every 4 epochs and report average over *top-5* mean success rates across all evaluation epochs. For task generalization (**Q3**) we evaluate the train policy on novel visual-semantic tasks not seen during training. For all evaluations we use 20 rollouts per task. Further training details are provided in Appendix C.2.1.

4.5 Experimental Results

First, we evaluate the effectiveness of our multi-resolution approach against common multi-task baselines, RT-1(63) and BC-Zero(64). We then present results for each research question. For qualitative results see: <https://sites.google.com/view/multi-res-real-time-control>.

4.5.1 Comparison to Multi-Task Baselines

Table 4.1 shows the results for the multi-task baselines RT-1(63) and BC-Zero(64) across all task. We note that for coarse manipulation tasks (MT-Coarse) these baselines, that use single camera views, can perform quite well. This is because these tasks only require coarse localization of the target object for task completion. However, for precise manipulation tasks (MT-Precise), such baselines perform quite poorly since these tasks require fine-grained grasping (as many objects are $\approx 1\text{cm}$ in size) and insertion for successful task completion.

On the other hand, our multi-resolution approach, performs much better as it uses the first-person camera view and force-feedback for finer grasping and insertion. For dynamic tasks (MT-Dynamic), our method considerably outperforms the baselines (1.5x). This is because dynamic tasks require *reactive* response to contact events.

	MT-Coarse	MT-Precise	MT-Dynamic
RT-1	81.0	12.5	4.5
BC-Z	74.1	7.8	4.8
Ours	82.0	55.0	73.6

Table 4.1: Task success comparison for multi-task baselines across all task domains.

	π_{-Ih}	π_{-I3}	π_{-FT}	Ours
MT-Coarse	74.5	41.0	81.8	82.0
MT-Precise	7.7	29.6	56.1	55.0
MT-Dynamic	65.8	27.5	33.2	73.6

Table 4.2: Results for multi-spatial resolution experiments (Section 4.5.2). Here, $-$ implies that we remove this input from policy. Thus, π_{-Ih} implies that the policy only operates on third-person camera views and force-torque feedback.

	$\pi_{\text{low-res}}$	$\pi_{\text{high-res}}$	Ours
MT-Coarse	82.0	81.0	82.0
MT-Precise	53.4	56.2	55.0
MT-Dynamic	4.2	12.2	73.6

Table 4.3: Results for multi-temporal resolution experiments (Section 4.5.2). Here, both $\pi_{\text{low-res}}$ and $\pi_{\text{high-res}}$ are single-resolution approaches which run at 5 Hz and 20 Hz respectively, while ours is a multi-resolution approach.

Only our multi-temporal resolution approach utilizes high spatial and temporal resolution sensing, enabling fast response to contact events.

4.5.2 Additional Baseline Comparisons

Q1 – Spatial Resolution Experiments: We now compare against the *spatial* resolution baselines discussed in Section 6.4.1. For this set of baselines all methods use multi-*temporal* resolution sensing with high-frequency force-torque feedback. Table 4.2 shows results across all task settings. For MT-Coarse we see that only using a first-person camera (π_{-I3}) performs poorly. This is because of partial observability in this view, i.e., the target object can be out of view and lead to task failure. On the other hand, for MT-Precise (Row 2), only using first-person camera (π_{-I3}) performs better ($\approx 2\times$) than using only the third-person camera (π_{-Ih}). This is because MT-Precise tasks require finer motions which are hard to perform from low spatial resolution (third-person) view only. Further, for dynamic tasks (Row 3), using first-person views alone again suffers because of partial observability.

Q2 – Temporal Resolution Experiments: Table 4.3 compares against single-temporal resolution baselines ($\pi_{\text{low-res}}$ and $\pi_{\text{high-res}}$). Table 4.2 shows that for coarse and precise domains single-resolution perform as well as our multi-resolution approach. This is because tasks in both domains are quasi-static and hence fast reaction to contact events is not critical for task success. On the other hand, for dynamic tasks (Table 4.2 bottom row), since fast response to contact events is necessary (to avoid failures such as object toppling, see Figure 4.5) our multi-resolution approach performs better than both $\pi_{\text{low-res}}$ (5Hz) and $\pi_{\text{high-res}}$ (20Hz) since it incorporates force feedback at 75Hz.

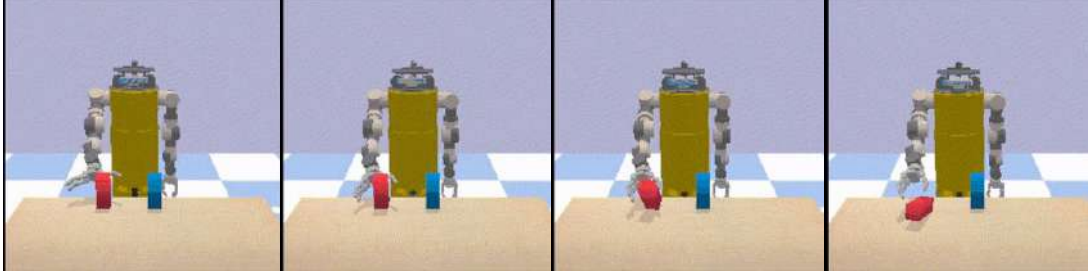


Figure 4.5: Example failure case for MT-Dynamic (Ballbot) task. As can be seen in the figure, if the robot approaches the object but does not react fast enough to the object contact, the block can topple resulting in task failure.

	$\pi_{I3-Frozen}$	π_{I3-FT}	$\pi_{multi-res-FT}$	Ours
MT-Coarse (Visual)	74.5 / 7.1	81.8 / 25.8	82.4 / 45.6	82.0 / 72.3
MT-Coarse (Geometry)	44.2 / 16.8	56.4 / 18.4	60.7 / 31.9	58.9 / 44.6
MT-Precise (Visual)	7.7 / 4.5	15.6 / 9.2	56.4 / 31.9	55.0 / 48.1

Table 4.4: Robustness experiment results, each cell shows *train/heldout* success rate (Section 4.5.2)

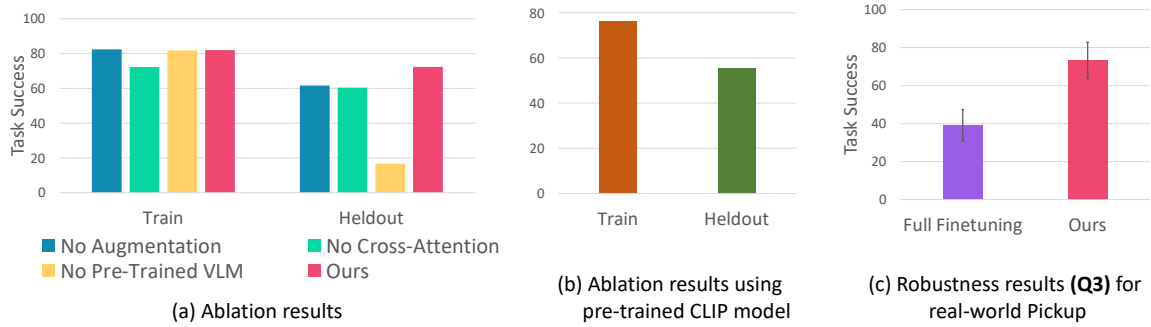


Figure 4.6: *Left:* Ablation results (see Section 4.5.3). *Right:* Robustness result for real-world pickup.

Q3 – Robustness Experiments: Table 4.4 compares results (*train / heldout*) for visual-semantic generalization against the robustness baselines in Section 6.4.1. As noted previously, for these experiments we evaluate the trained policies on *heldout* environments (see Appendix C.2.1 for details). We note that our approach, with frozen pretrained model, generalizes better than the finetuned model $\pi_{\text{multi-res-FT}}$. This shows the ability of our approach to maintain the generalization capabilities of the pretrained VLM as compared to the finetuned model that suffers from ‘forgetting’ and representation drift towards the training tasks. Additionally, from column-1 and column-2, we again note that the finetuned $\pi_{\text{I3-FT}}$ model suffers a larger decrease in performance as compared to $\pi_{\text{I3-Frozen}}$. Finally, comparing $\pi_{\text{I3-FT}}$ against $\pi_{\text{multi-res-FT}}$, we see that even with finetuning our multi-spatial resolution approach generalizes better because it can utilize first-person views for improved task success.

Real-World Experiments: We evaluate our approach in the real-world on two tasks, pickup and peg-insertion (35). Table 4.5 shows comparison against the spatial resolution baselines. We note that our approach, with multi-spatial resolution, performs $\approx 3\times$ better than the baselines on both tasks.

We see that given *limited* demonstrations both π_{I3} and π_{Ih} fail to perform well (across both tasks). On the other hand, removing force-torque feedback π_{Ih} only affects performance on insertion task ($\approx 20\%$ less) since this task relies more on contact feedback. Additionally, Figure 4.6 (c) figure plots the robustness result for pickup task. As before we see that our approach with frozen model performs better. See website for qualitative results.

	π_{Ih}	π_{I3}	π_{FT}	Ours
Pickup	7.5 (3.5)	20.0 (14.1)	67.5 (3.5)	75.0 (7.0)
Peg-Insert	10.0 (0.0)	12.5 (4.6)	42.5 (3.5)	67.5 (3.5)

Table 4.5: Mean (stdev) results (using 2 seeds) for multi-spatial resolution for real world tasks.

4.5.3 Ablations

We further ablate the different components of our proposed approach. Due to space limitations we only summarize key findings and provide details in Appendix C.3.2

Pixel-Level Augmentations: We evaluate the effect of pixel-level augmentations (color jitter, gray-scale) on the training and generalization of our MT-policies on MT-Coarse. Figure 4.6 reports results on both training and heldout (novel) evaluation configurations. We see that while there is very little difference in training performance, extensive pixel-level augmentations helps generalization by close to $\approx 15\%$. While pixel-level augmentations change the semantics of the task, our multi-modal approach is still able to complete the task because of visual-language grounded provided from pretraining.

Spatial Sensor Fusion using Cross-Attention: We compare use of early fusion using cross-attention with late fusion using concatenation. Figure 4.6 (a) (green

bar) shows that using cross-attention improves the performance by around $\approx 8\%$ on both train and heldout configuration. Thus, using cross-attention for multi-modal fusion is more effective than concatenation. However, we note that cross-attention requires more parameters and has slower inference.

Effect of Pretraining: We also evaluate the effects of using pretrained-VLMs. Figure 4.6 (a) (yellow bar) shows the training and heldout performance using ImageNet initialization which only has visual pretraining and no vision-language pre-training. We see that while training performance matches our approach the heldout performance decreases tremendously. This large decrease is due to missing visual-language grounding since we use *separately* trained visual and language models. We also evaluate the effects of using pretrained-VLMs.

4.6 Conclusions and Limitations

Our work proposes using sensing modalities at multiple spatial and temporal resolutions for learning multi-task manipulation policies. Our multi-resolution approach captures information at multiple hierarchies and allows the robot to perform both coarse and fine motions with high reactivity and real-time control. To learn generalizable multi-task policies we further leverage off-the-shelf pretrained vision-language models and freeze them to maintain their robustness. Our work has several limitations. While our proposed framework is general for multi-spatial sensing we only rely on global third-person camera and local first-person camera view. Further local sensing using vibro-tactile sensors (121; 122; 123) was not explored. Further, it is unclear if our approach of using cross-attention for sensor fusion will be optimal for more than 2 sensors. Additionally, while our multi-resolution policy allows us to learn robust policies not all sensing modalities will be available for all tasks. Thus, future work should explore adapting to scenarios with missing sensing modalities.

4.7 Acknowledgments

We gratefully acknowledge support for this work from the National Science Foundation under Grant No. CMMI-1925130 and IIS-1956163. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

5

VLTSafe: VISION-LANGUAGE GUIDED SAFE DYNAMIC MANIPULATION VIA OBJECT-CENTRIC TRANSFORMERS

Safely executing dynamic manipulation tasks—like pulling a box from under a stack—in cluttered, open-world environments is challenging. Robots must identify implicit safety constraints (e.g., avoid collapsing stacks or hitting fragile items) while simultaneously reasoning about the combinatorially large number of interactions and their long-term safety consequences (e.g., aggressively pulling an object from under a stack will cause it to fall and hit nearby objects). In this work, we present *VLTSafe: Vision-Language guided Transformers for Safe manipulation*, a framework for safe dynamic manipulation that leverages vision-language models (VLMs) to translate semantic safety concepts into geometric constraints, and object-centric transformers to learn generalizable low-level safe policies. To tractably learn policies that scale well to complex cluttered settings, we: (i) utilize a transformer architecture, representing objects as tokens, enabling a *single* policy to be deployed in variable degrees of clutter; (ii) consider diverse combinations of constraint types during training, enabling generalization to novel test-time constraint compositions; and (iii) optimize a reach-avoid reinforcement learning objective to train a parameterized policy that reasons about long-term safety as well as task completion. At test time, *VLTSafe* uses a VLM to identify relevant geometric constraints from RGB images and a textual task description, enabling open-world constraint specification. In both simulation and hardware experiments with a Franka Panda arm, *VLTSafe* infers nuanced constraints and goals (e.g., soft loofahs can safely be pushed out of the way) that cannot be easily identified with hand-designed heuristics. Furthermore, the learned safe policy shows zero-shot generalization to highly cluttered scenes and novel constraint compositions owing to the transformer’s time-varying attention over *relevant* objects.

5.1 Introduction

Consider a robot manipulator that must carefully pull a box from under a stack while surrounded by clutter on a table, then place the box on the far side of the table away from any fragile objects. Accomplishing this task safely requires the robot to reason about many types of interactions and constraints, some of which are directly influenced by the dynamics of the interaction (the top boxes can be flipped over or thrown off the table if the bottom box is pulled aggressively), and others that are semantically informed by the semantic properties of interacting objects – making gentle contact with fragile objects but freely pushing soft objects out of the way to complete the task. All of these interactions must be accounted for, with their corresponding safety constraints respected, while completing the task of pulling a box from under a stack and placing it at a goal location. However, as the environment becomes more complex, explicitly modeling all possible interaction combinations quickly becomes intractable.

In this work, we seek to compute low-level control policies that can safely perform dynamic manipulation tasks in arbitrary cluttered environments and with nuanced robot-object interaction constraints. The core challenge lies in *tractably* computing a policy which preserves *test-time generalization* to novel constraint compositions, object configurations, and degrees of clutter. To tackle this, we introduce **VLTSafe**, framework for safe dynamic manipulation that leverages vision-language models (VLMs) as a test-time constraint specifier and a sim-to-real training recipe for learning generalizable low-level safe policies.

Specifically, **VLTSafe** builds on an object-centric transformer policy architecture, where each object in the scene is represented as a token enriched with constraint attributes derived from a vision-language model’s (VLM) analysis of the scene. We design a custom attention mask based on the key insight that, in cluttered environments, only a small subset of objects are relevant for safe decision-making at any given time. This attention mechanism guides the transformer to focus on critical interactions without being ‘distracted’ by irrelevant context. We train this parameterized policy entirely in simulation using a reach-avoid reinforcement learning (RL) objective (124), which allows the model to reason about satisfying long-horizon safety constraints while pursuing the task goal. During training, we systematically vary both the level of clutter and the composition of constraints, encouraging the policy to learn how different object interactions impact long-term safety across a wide range of scenarios. The result is a single transformer-based policy that generalizes to scenes with varying numbers and types of objects and constraints, without requiring retraining. At test time, given a single image and a task description, an off-the-shelf VLM identifies relevant safety constraints (e.g., “do not touch fragile objects” or “avoid toppling the stack of boxes”) and goal specifications (e.g., “place the box away from fragile objects”). These high-level instructions are translated into geometric constraint parameters, which are then used to condition the transformer policy for deployment in

novel environments.

We evaluate **VLTSafe** in controlled simulation experiments and we zero-shot deploy the policy from sim to real on a 7-DoF Franka Panda arm performing the motivating dynamic manipulation task. We find that **VLTSafe** achieves strong zero-shot generalization to novel constraint compositions and significantly higher levels of clutter than those seen during training. It consistently outperforms baseline methods, achieving higher safe success rates and lower failure rates. Our custom attention masking proves crucial for generalization, particularly due to its ability to focus attention on the most relevant objects in cluttered scenes. Moreover, training with diverse constraint types further improves robustness, enabling effective deployment to novel constraint compositions without the need for retraining.

5.2 Related work

Safe Control for Robotic Manipulation Impedance and null space control are widely used to ensure compliant interactions in robotic manipulation. Impedance control modulates stiffness and damping properties to enable safe and adaptive interactions with the environment (125; 126), while null space control allows secondary objectives like collision avoidance to be incorporated without interfering with primary tasks (127; 128). In contrast, contact-aware controllers explicitly reason about contact interactions to keep interaction forces below a safety threshold (129; 130). Recent work has focused on learning contact-aware compliant controllers using expert demonstrations (131; 132) and RL (133; 134; 135). Our work builds on these efforts by enabling contact-aware control through reachability-based methods in cluttered environments.

Reachability-based Safe Control Hamilton-Jacobi reachability analysis (136; 137; 138) provides theoretical formulations for finding the solution to nonlinear reach-avoid control problems by minimizing the worst case (minimum over time) loss. Compared to approaches that minimize the cumulative loss over time (139; 140), HJR analysis provides rigorous safety assurances, ensuring that the system avoids unsafe states under worst-case scenarios. However, HJR becomes computationally intractable as the dimension of the state space increases (141). Recent methods (124; 142; 143) take inspiration from RL-based approaches (144; 145) to extend reachability analysis to higher dimensions using a discounted formulation of the reach-avoid bellman equation. Leveraging HJR analysis for learning safe policies in dynamic, cluttered manipulation remains underexplored due to the high dimensionality of complex multi-body interactions.

Semantic Safe Planning using Vision-Language Models Recent works have explored incorporating semantic safety into task planning. (146) integrate safety prompts in the code-as-policies (147) setup, while (148) use LLMs to decompose high-level tasks into subtasks and verify them against LTL specifications. Other

approaches infer user preferences through demonstrations and active queries (149). Semantic constraints derived from VLMs have been used for planning, with simulators verifying feasibility (150; 151). However, these approaches primarily focus on quasi-static tasks without long-horizon reasoning. Most relevant to our work is (152), which incorporates dynamic constraints within a formal safety framework. However, it requires manually designing barrier functions for each constraint type, and the policy accounts for all constraints simultaneously, making it impractical for cluttered environments. Our approach leverages RARL to learn safety value functions that can be solved for both *liveness* and *safety* based on a subset of relevant task and safety constraints.

5.3 Problem Formulation and the *VLTSafe* Method

In this work, our goal is to compute low-level robot policies that can safely perform dynamic manipulation tasks in cluttered environments with nuanced object interaction constraints. We first formalize the problem mathematically, revealing the underlying challenges. We then present our approach—***VLTSafe***—which breaks down these challenges by training an object-centric transformer policy that is informed about relevant safety constraints at test-time by a vision-language model. Our proposed approach is shown in Fig. 5.1.

Notation

We model the robot’s state as the position and velocity of the end-effector $s^{\text{EE}} = [x^{\text{EE}}, \dot{x}^{\text{EE}}]$ where $x^{\text{EE}} \in \mathbb{R}^3$ is the Cartesian position and $\dot{x}^{\text{EE}} \in \mathbb{R}^3$ is the velocity. Let each object’s state be represented by $s_i^o = [x_i^o, \dot{x}_i^o], i \in \{1, \dots, N\}$ where N is the total number of objects in the cluttered scene. We denote the *full state* of the robot and the objects via $s = [s^{\text{EE}}, s_1^o, \dots, s_N^o] \in \mathcal{S}$ where \mathcal{S} is the full state space. Finally, let the robot’s actions $a \in \mathcal{A}$ be represented as the Cartesian displacement of the end effector $\Delta x^{\text{EE}} \in \mathbb{R}^3$. In this work, we do not assume access to an analytic dynamics model $s_{t+1} = f(s_t, a_t)$, but instead rely on a high-fidelity simulator (153) to evolve the robot and object states as a result of the robot’s actions and physical interaction.

Safety Constraint & Task Target Representation We represent safety constraints as a *failure set*, $\mathcal{F} \subset \mathcal{S}$, which encodes the forbidden robot-object, object-object or object-environment states. For example, the failure set can prohibit high-velocity contact with fragile objects while permitting arbitrary interactions with soft objects. We seek robot policies that not only comply with this safety specification but are also guaranteed to complete a task. Let the *target set*, $\mathcal{T} \subset \mathcal{S}$, be the set of all states that satisfy the robot’s task (i.e., reach a goal). For example, placing a book on an empty shelf.

Goal: A Safe Dynamic Manipulation Policy Adaptable to Test-time Objects and Constraints Our goal is to find a low-level robot policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which respects the safety constraints \mathcal{F} while also achieving the desired task target \mathcal{T} . To

enable test-time adaptation to relevant constraints and targets in the deployment environment, we parameterize the policy via the constraints and targets, denoted by $\pi_{\mathcal{F}, \mathcal{T}}$. Formally, we model this problem as a reach-avoid problem (124; 137; 138). In order to compute the policy which satisfies both properties, we pose a Hamilton-Jacobi reachability problem (136). Here, the failure set is encoded via the function $g(\cdot; \mathcal{F})$ and the target set via $\ell(\cdot; \mathcal{T})$ where

$$g(s; \mathcal{F}) < 0 \iff s \in \mathcal{F}, \quad \ell(s; \mathcal{T}) \geq 0 \iff s \in \mathcal{T}.$$

To jointly account for *multiple* constraints, we define $g(s; \mathcal{F}) := \min(g_1(s), \dots, g_q(s))$ where q is the number of safety constraints and to account for *multiple* task goals, we define $\ell(s; \mathcal{T}) := \min(\ell_1(s), \dots, \ell_p(s))$ where p is the number of task objectives. Note that both of these functions must be parameterized by the relevant constraint or goal specification.

Following (124; 137), we formulate our policy optimization problem as:

$$V(s; \mathcal{F}, \mathcal{T}) := \max_{\pi_{\mathcal{F}, \mathcal{T}}} \left\{ \max_{t \geq 0} \min \left\{ \ell(\xi_s^\pi(t); \mathcal{T}), \min \{g(\xi_s^\pi(t); \mathcal{F})\} \right\} \right\}. \quad (5.1)$$

where ξ_s^π is a state trajectory starting from state s and executing π , and we index into this trajectory at any discrete time via (t) notation. The inner optimization (starting with $\max_{t \geq 0}$) “remembers” if the robot was able to reach the target \mathcal{T} without violating the constraint \mathcal{F} . The value function will be positive only if both reaching and avoiding happen successfully. The policy’s goal is to maximize this objective, hence reaching the goal without ever violating the specified constraints.

Challenge: Tractable Computation while Preserving Test-Time Generalization Solving the above optimization over all possible objects, tasks, and safety constraints is intractable. While restricting to a predefined subset simplifies the problem, it limits safe deployment in novel environments. Therefore, we need an approach that generalizes safe policies across diverse objects and constraint compositions to enable safe deployment in open-world, cluttered environments.

5.3.1 Offline: Safe Policy Learning via Object-Centric Transformers & Reach-Avoid RL

Policy Representation: Masked Object-centric Transformer Architecture

We propose an object-centric transformer-based model (154) as our policy representation (Fig. 5.2). This architecture offers two key advantages. First, its input representation is flexible, supporting a variable number of objects and constraint types at test time. Second, by structuring the transformer’s attention mask we can control how the policy attends to objects in extremely cluttered scenes. Our aim is to learn a policy $\pi_{\mathcal{F}, \mathcal{T}}$ parameterized via constraints and targets, hence we concatenate the input observations with the parameters of the failure ϕ and target τ sets, such that

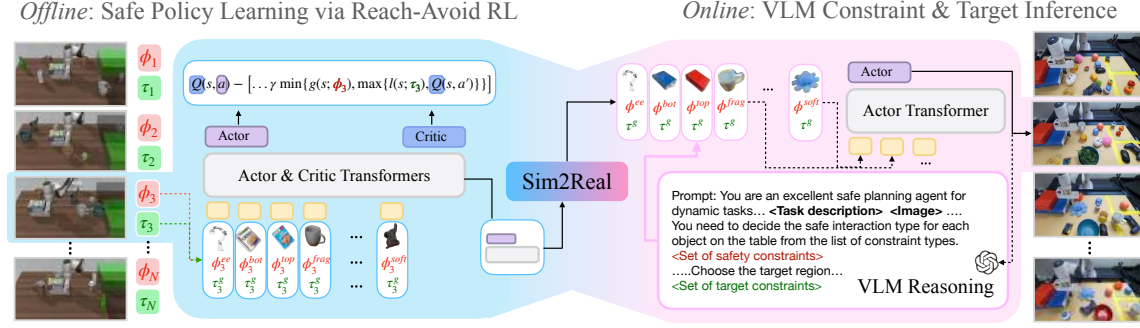


Figure 5.1: **Method Overview:** (left) In simulation, we train a parameterized object-centric transformer policy using reach-avoid reinforcement learning with diverse environments, tasks, and safety constraints. (right) At test time, given an RGB image, task description, and a constraint vocabulary, a vision-language model (VLM) infers the relevant semantic goals and safety constraints for each object. Our training recipe enables zero-shot sim2real transfer and generalization to novel constraint compositions in highly cluttered scenes.

state of each object becomes $[s, \phi, \tau]$. These observations are tokenized through a two-layer MLP and subsequently processed using two layers of multi-head self-attention, where the attention mechanism is guided by a *custom attention mask*. We design our custom attention mask based on a key insight: in cluttered environments, only a small subset of objects are "relevant" for safe decision-making at any given time step. Consequently, our custom attention mask prioritizes attention to the most relevant neighboring objects. Since we use multiple attention layers, the mask does not fully block information from more distant objects—it simply biases attention toward closer ones. Specifically, all object tokens attend to the end-effector tokens and vice versa, while object-to-object attention is restricted to their k -nearest neighbors, where k is task-dependent and can vary across objects (e.g., in our experiments, we choose $k = 3$), while also allowing objects in the stack to always attend to one another.

Policy Optimization: Reach-Avoid Reinforcement Learning With our policy architecture setup, we now turn to policy optimization. Recall the reach-avoid optimization problem we formulated in Equation 5.1. Exact grid-based dynamic programming solvers are intractable for the high-dimensional state space representation we use in this work. Thus, to tractably compute a reach-avoid policy, we use a reinforcement-learning based relaxation of this optimization. Specifically, we adopt the time-discounted reach-avoid Bellman backup from (124) which discounts the safety value function to induce a contraction mapping, yielding the reach-avoid problem compatible with off-the-shelf reinforcement learning solvers.

$$Q(s, a; \mathcal{F}, \mathcal{T}) = (1-\gamma) \min \left\{ \ell(s; \mathcal{T}), g(s; \mathcal{F}) \right\} + \gamma \min \left\{ g(s; \mathcal{F}), \max \left\{ \ell(s; \mathcal{T}), Q(s', a'; \mathcal{F}, \mathcal{T}) \right\} \right\}$$

where $\gamma \in [0, 1)$ is the discount factor, which can be interpreted as the probability of episode continuation. We utilize Deep Deterministic Policy Gradient (DDPG)

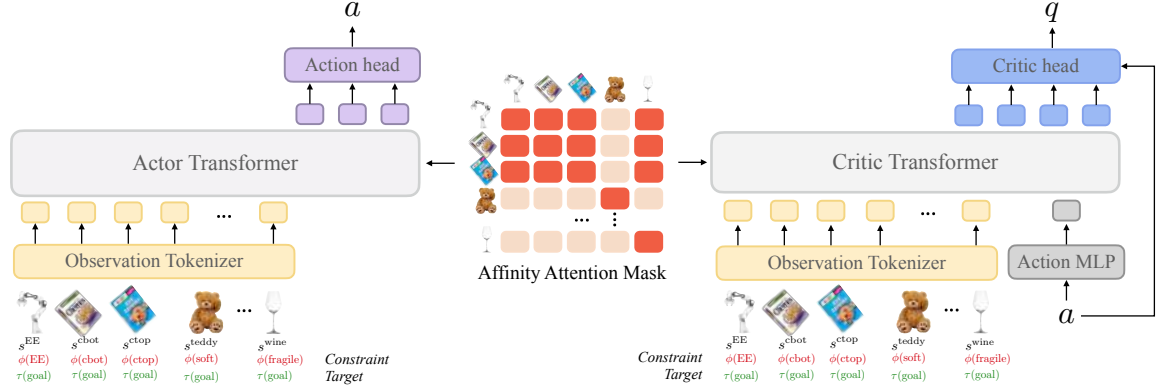


Figure 5.2: **VLTSafe System Architecture:** Actor and critic networks are object-centric transformer-based models. Observations are augmented with parameters of the target τ and failure set ϕ and tokenized. These tokens are processed using multi-head self-attention with a custom attention mask. Output tokens are aggregated and used to output action (actor) and value estimate (critic).

(155) as our RL framework wherein the bellman update is modified to incorporate the time-discounted reach-avoid Bellman backup (Equations in Appendix D.1.) We model both the actor and critic using object-centric transformer architectures (Fig. 5.2). The actor network encodes observation tokens with a transformer, whose output tokens are aggregated via mean pooling and passed through a 2-layer MLP policy head to produce the next action. Critic network jointly encodes state and action tokens using a separate transformer with two layers of multi-head self-attention. The resulting token embeddings are aggregated and concatenated with the action, then passed through a two-layer MLP value head to estimate the Q-value.

5.3.2 *Online: Adapting via Safety Constraints Inferred from Vision-Language Models*

The prior section allowed us to pre-compute a low-level robot policy which given the current state of the scene s , a safety specification \mathcal{F} , and a goal specification \mathcal{T} safely performs the task. The question is, at deployment time, how can the robot know what are the “relevant” safety constraints \mathcal{F} and how they may impact which goals \mathcal{T} are and aren’t allowed? Here, we propose using a vision-language model (VLM) as an open-world constraint and target specifier. Given the current image of the scene \mathcal{I} and a text prompt describing the task \mathcal{L}_{task} , the VLM should use the visual and semantic cues to select the relevant safety and targets that are passed into the low-level policy.

However, a large challenge with using VLMs is the right interface between textual representation (that the VLM is trained to output) and the embodied representation (e.g., robot states, objects, etc.) that our robot policy needs at test-time to

adapt. To bridge this gap, we model the selection of constraints/targets as a multiple choice visual question-answering (VQA) problem where the multiple choice options are generated via pre-defined geometric constraint and target functions. In the future, we envision the possibility of these functions to be written by another LLM based model (e.g., as in Code-as-Policies (147)). We first convert the set of all geometric constraints \mathcal{T}, \mathcal{F} we can generate with our functions to corresponding textual descriptions $\mathcal{L}_{\mathcal{F}}, \mathcal{L}_{\mathcal{T}}$. For example, a hard collision-avoidance constraint between two interacting bodies is represented in text as:

$$\text{Math Representation of } \mathcal{F}: \quad g(s_{\text{EE}}, s_{\text{cup}}) = ||x_{\text{EE}} - x_{\text{cup}}|| - \epsilon \quad (5.2)$$

$$\text{Text Representation of } \mathcal{F}: \quad \mathcal{L}_{\mathcal{F}}(\langle \text{EE} \rangle, \langle \text{cup} \rangle) = \langle \text{no contact} \rangle \quad (5.3)$$

For the target set, the geometric objective placing a book on an empty book shelf can be written as:

$$\text{Math Representation of } \mathcal{T}: \quad \ell(x_{\text{book}}) = (||x_{\text{book}} - x_{\text{shelf}}|| - \epsilon \quad (5.4)$$

$$\text{Text Representation of } \mathcal{T}: \quad \mathcal{L}_{\mathcal{T}}(\langle \text{book} \rangle) = \langle \text{second shelf} \rangle, \quad (5.5)$$

if the VLM detects the second shelf to be empty. Note that without a VLM, such open-world specifications would need to be manually specified by an expert designer. Once the set of semantic constraints are selected by the VLM, they can be converted back to their geometric formulations and used to parameterize the learned policy for execution. Constraints are identified at the beginning of each episode ($t = 0$) and kept fixed for the duration of the rollout.

5.4 Simulation Experiments

We first devised a series of simulation experiments to carefully test the impact of each of our design decisions on overall task performance. Specifically, we ask the questions (1) When the training and test environments are the same, how much does the transformer architecture influence task success?, (2) How well can our approach generalize to varying degrees of novel, test-time object clutter?, and (3) How well does our approach generalize to novel, test-time constraint compositions?

5.4.1 Experimental Setup

Environment & Task In all experiments, we use a tabletop setup where a Franka robot interacts with household objects (Fig. 5.5). The task involves quickly but safely pulling a cereal box from under another box on a cluttered tabletop and placing it in a goal region. We simulate the robot and environment dynamics with MuJoCo (153) and use the Google Scanned Objects Dataset (156; 157; 158) to simulate everyday objects in the table arena defined in Robosuite (159).

Safety and Target Representation

Intuitively, the safety constraints capture how the stack of boxes should remain stable as the robot pulls the cereal box from underneath, modeling critical robot-object interactions. Specifically, \mathcal{F} enforces that the top cereal box does not displace beyond a threshold, the end-effector avoids contact with *fragile* objects, may make limited-velocity contact with *soft* objects, can freely interact with *durable* objects, and must not move over *sensitive* items like a laptop. Since these constraints depend on the semantic properties of objects, the VLM assigns constraint types via the parameterization $\phi \in \{\text{no-contact, soft-contact, any-contact, do not move over}\}$. The target set \mathcal{T} defines successful task completion as the bottom cereal box being fully separated from the top box and placed in a goal region at the far end of the table. This yields two possible targets – **top-goal** and **bottom-goal** – which the VLM selects at deployment time via the parameterization $\tau \in \{\text{top-goal, bottom-goal}\}$. Additional implementation details are provided in Appendix D.2.

VLM for Test-Time Safety and Target Specification We use GPT-4o (160) as the VLM. It is queried once at the beginning of a test episode to identify the constraint types relevant to all objects in the scene and to select the target constraint that is most appropriate for the task. The full prompt for the constraint selection is provided in Appendix D.3

Metrics We measure (1) *safe success rate* (SafeSucc %) defined as the percentage of trajectories that complete the task safely (i.e., satisfy reach and avoid), (2) *stack safety violation rate* (StackFail %) defined as the top cereal block moving outside of safety limits, and (3) *object safety violation rate* (ObjFail %) percentage of objects not in the stack that are unsafely interacted with.

Training details Model architectures and hyperparameters are provided in the Appendix D.2).

Method	Always Same Constraint (<i>AllFragileConst</i>)			Random Constraints per Episode (<i>RandomConst</i>)		
	SafeSucc (%) \uparrow	StackFail (%) \downarrow	ObjFail (%) \downarrow	SafeSucc (%) \uparrow	StackFail (%) \downarrow	ObjFail (%) \downarrow
MLP	82	9	9	40.5	43.5	15.5
VLTSafe-NoMask	85.5	3.5	8.5	60.3	29.6	10.1
VLTSafe	95.5	0.5	4	78.5	11.5	4

Table 5.1: **Policy Architecture vs. Performance.** Comparison of our method against baselines in two training domains, *AllFragileConst* and *RandomConst*

5.4.2 How much does the transformer policy architecture influence task success?

First, we study how much the transformer design influences the overall task success, assuming the training and test environments are the same.

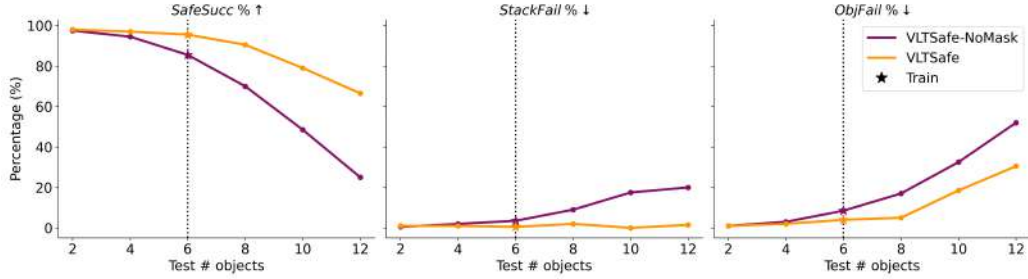


Figure 5.3: **Generalization: Varying Degrees of Clutter.** Performance variation as the number of objects is varied in the test scenario. Both **VLTSafe** vs **VLTSafe-NoMask** are trained in the *AllFragileConst* domain with six objects. Black dotted line represents the number of objects seen during training.

Methods We compare three policy architectures: **VLTSafe** which is our object-centric transformer architecture *with* masking, **VLTSafe-NoMask** which has no masking, **MLP** which is a MLP policy architecture. To ensure a fair comparison, we kept the representation capacity (number of trainable parameters) roughly the same across all models (see Appendix D.2 for details).

Training & Test Setup We train each policy in two training domains. First, we train policies in the *AllFragileConst* domain where all objects are assigned the same constraint type during all episodes: $\phi = \text{no-contact}$ (i.e. all objects are assumed to be fragile). Next, we train policies in the *RandomConst* domain. Here, constraints for each object are randomly sampled (from the set of constraints) at the start of each episode during training. We train with only six objects in the scene and then deploy with six objects in the scene, and the training and the test domain are identical.

Results We report all metrics described Sec. 5.4.1 in Table 6.1. We find that **VLTSafe** outperforms both baselines with the highest safe success rate and lowest failures rates. Specifically, we find that the use of masking significantly helps the policy. We hypothesize that, without masking, the transformer attends equally to all objects in the scene, making the learning problem unnecessarily difficult. By restricting attention to only the most “relevant” objects through masking, the model can focus on critical near-term interactions – such as between the two cereal boxes – enabling more effective learning of safer policies. This effect is further highlighted in *RandomConst*, where **VLTSafe-NoMask** performs noticeably worse than **VLTSafe**. Furthermore, even with the same representation capacity, the MLP model is unable to learn safe policies that satisfy complex interactive constraints. This further highlights the advantage of using a transformer-based architecture.

5.4.3 How well can our approach generalize to varying degrees of test-time clutter?

Methods Next, we study the ability of the policy to generalize to novel object/clutter amounts at test-time. Since the MLP cannot handle arbitrary number of object at test-time, we only compare **VLTSafe** and **VLTSafe-NoMask**.

Training & Test Setup During training, we only consider the *AllFragileConst* domain trained with six objects, as in Section 5.4.2. At test-time, we deploy each policy in varying degrees of clutter—with 2,4,6,8,10, and 12 objects—to test generalization to number of scene objects.

Results In Fig. 5.3, our quantitative results show that masking plays a large role in enabling the policy to maintain performance and generalize to environments with varying degrees of clutter. Specifically, with **VLTSafe** we see a 29% decrease in the safe success rate when we double the number of objects from 6 to 12, while with **VLTSafe-NoMask** we see a 60.5% decrease. Looking deeper into the failure modes, we see that the main gains come from how **VLTSafe** maintains a near zero-percent StackFail rate as we increase the number of objects. In contrast, for **VLTSafe-NoMask** the StackFail rate increases to 20%. We hypothesize that, similar to 5.4.2, this is again due to the capability of **VLTSafe** to focus on relevant objects and critical near-term interactions without being ‘distracted’ by the apparent complexity of the task.

5.4.4 How well does **VLTSafe** generalize to novel, test-time constraint compositions?

Methods Next, we want to study how well our approach generalizes to novel constraint compositions at test-time. Here we only test **VLTSafe**.

Training & Test Setup

We train our policy in two settings: *RandomConst*, where constraints vary randomly between episodes, and *FixedConst*, where constraints are fixed for each object. Training is limited to 6-object scenes, and at test time, we evaluate generalization in the *RandomConst* domain with 6, 8, and 10 objects and novel constraint combinations.

Results Recall that at training-time (with six objects), **VLTSafe** achieves a safe success rate of 79.5% in the *FixedConst* domain and a safe success rate of 78.5% in *RandomConst* (star icon in Fig. 5.4). Note that this test scenario is novel for the policy trained in *FixedConst* even with N=6, since during training the policy never saw randomized object constraints. This explains the drop in performance (12.5 %) when this policy is tested in *RandomConst*. For the policy trained with *RandomConst*, only the 8 and 10-object environments are novel and, since the number of objects is larger, the constraint composition is novel. In Fig. 5.4, we see that **VLTSafe** trained with random combinations of constraints exhibits better generalization capabilities,



Figure 5.4: **Generalization: Constraint Compositions.** Evaluating policies trained in *FixedConst* and *RandomConst* domains with six objects, in *RandomConst* domain with increasing number of objects.

compared to training with *FixedConst*, since it sees a larger variation in constraint compositions during training. However, we note that this performance gap quickly diminishes as the environment complexity increases and the task becomes significantly more challenging than the train setting.

5.5 Real World Experiments

Hardware Setup and State Estimation We use a 7-DoF Franka Emika Panda arm and an Azure Kinect RGB-D camera for both state estimation and image capture for VLM-based constraint inference. State estimation relies on TapNet (161; 162), a point tracking algorithm applied to RGB-D inputs. At the start of each trajectory, we manually annotate object keypoints, a step that could be automated in future work using VLMs. TapNet tracks object states at 17 Hz.

Sim-to-Real We deploy policies trained entirely in simulation directly in the real world, without any real-world fine-tuning. The experimental setup including the robot’s state space and the object configurations on the table closely matches that of the simulation environment, enabling zero-shot transfer of policies trained in simulation using object-centric observations, directly in the real world. We deploy the policy trained in the *RandomConst* simulation domain. This policy was trained considering six objects in the scene, however we test in scenes containing up to eight objects.

Results Fig. 5.5 shows a real world rollout of our learned policy in a scene with eight objects in clutter. First, a VLM identifies the goal and object constraints: Soft objects are typically assigned the `<any contact>` constraint, while mugs and electronic items considered safety-critical are labeled as fragile `<no contact>`. When prompted to choose between two target sets — one containing soft toys, and the



Figure 5.5: **Real-world deployment:** Zero-shot sim-to-real transfer of a policy trained in simulation (*RandomConst*, 6 objects) to a real-world setting with 8 objects. Constraints are first inferred using a VLM. As the robot pulls the blue box from under the red one, it makes allowable contact with the blue plush toy, then lifts to avoid the bowl and places the box safely in the goal region, away from the fragile porcelain mug.

other containing a porcelain mug — the VLM selects the former, as interactions with soft objects are safer. During execution, the robot briefly contacts the blue plush toy while extracting the blue box – an acceptable interaction then lifts the end-effector to avoid colliding with a nearby bowl. It ultimately places the box safely atop the orange loofah and blue ball, both of which are deemed safe for contact. Additional real world experiments are included in the Appendix D.4.

5.6 Conclusion

We introduce **VLTSafe**, an approach for learning low-level policies to safely perform dynamic tasks in cluttered environments. By leveraging vision-language models (VLMs) for open-world constraint synthesis, our object-centric transformer-based policy demonstrates zero-shot generalization to high degrees of clutter and novel constraint compositions. Training with the reach-avoid reinforcement learning (RARL) objective enables long-horizon safety reasoning and task accomplishment. **VLTSafe** outperforms baseline approaches, achieving high safe success rates, low failure rates, and strong generalization across novel scenarios.

5.7 Limitations

Our approach has several limitations. It relies on off-the-shelf state estimation methods to enable learning of object-centric representations. While this enables strong generalization and seamless sim-to-real transfer, it may limit performance in certain scenarios. Future work will explore visual object proposals, which can implicitly capture both geometric and semantic information, reducing reliance on traditional state estimation methods. Additionally, our experiments were constrained to the Cartesian state space, but we aim to extend our approach to 6-DoF manipulation for more dexterous and dynamic tasks.

6

GRAPHEQA: USING 3D SEMANTIC SCENE GRAPHS FOR REAL-TIME EMBODIED QUESTION ANSWERING

In Embodied Question Answering (EQA), agents must explore and develop a semantic understanding of an unseen environment to answer a situated question with confidence. This problem remains challenging in robotics, due to the difficulties in obtaining useful semantic representations, updating these representations online, and leveraging prior world knowledge for efficient planning and exploration. To address these limitations, we propose GraphEQA, a novel approach that chapter utilizes real-time 3D metric-semantic scene graphs (3DSGs) and task relevant images as multi-modal memory for grounding Vision-Language Models (VLMs) to perform EQA tasks in unseen environments. We employ a hierarchical planning approach that exploits the hierarchical nature of 3DSGs for structured planning and semantics-guided exploration. We evaluate GraphEQA in simulation on two benchmark datasets, HM-EQA and OpenEQA, and demonstrate that it outperforms key baselines by completing EQA tasks with higher success rates and fewer planning steps. We further demonstrate GraphEQA in multiple real-world home and office environments. Videos and code: [grapheqa.github.io](https://github.com/GraphEQA).

6.1 Introduction

Embodied Question Answering (EQA) (163) is a challenging task in robotics, wherein an agent is required to explore and understand a previously unseen environment sufficiently well, to answer an embodied question in natural language. Accomplishing this task efficiently requires agents to rely on both commonsense knowledge of human environments as well as ground its exploration strategy in the current environment context. For example, to answer the question “How many chairs are there at the dining table?”, the agent might rely on commonsense knowledge to understand that dining tables are often associated with dining rooms and dining rooms are usually near the kitchen towards the back of a home. A reasonable navigation strategy

would involve navigating to the back of the house to locate a kitchen. To ground this search in the current environment, however, requires the agent to continually maintain an understanding of where it is, memory of where it has been, and what further exploratory actions will lead it to relevant regions. Finally, the agent needs to observe the target object(s) and perform visual grounding to reason about the number of chairs around the dining table, and confidently answer the question correctly.

Maintaining a concise and effective memory and using it to take actions in the environment is critical for solving EQA tasks. Prior works have demonstrated the impressive commonsense reasoning capabilities of Vision Language Models (VLMs) as planning agents, while leveraging a semantic map for retrieval (164) or semantic exploration (165). In such approaches, the VLMs are not grounded in the current environment, and commonsense reasoning and context-based decision-making are disconnected. Recent works (166; 167; 168; 169; 170) focus on maintaining memory modules that can be queried by VLM agents for grounded planning. To construct a semantically rich memory, prior works either maintain a large, extensive set of images (170; 171) or have to perform an expensive offline processing step to obtain a compact representation (168; 169; 172). Thus, such semantic memory modules are either semantically rich (164; 169; 173), compact (166; 167), *or* online (167), but not all at the same time.

To address these limitations, we propose GraphEQA, an approach for embodied question answering that builds an online, compact, multimodal semantic memory combining global, semantically-sparse, and task-agnostic information from real-time 3D scene graphs (174) with local and semantically-rich information from task-relevant images (172). GraphEQA uses this multimodal representation for grounding vision-language planners to solve EQA tasks in unseen environments. Specifically, we utilize a recent spatial perception system (174) that incrementally creates a real-time 3D metric-semantic hierarchical scene graph (3DSG), given sequential egocentric image frames. We further augment this scene graph with semantic room labels and semantically-enriched frontiers, while maintaining a task-relevant visual memory that keeps track of task-relevant images as the robot explores the environment. Finally, we employ a hierarchical planning approach that utilizes the hierarchical nature of scene graphs and semantically relevant frontiers for structured planning and exploration in an unseen environment before using the multimodal memory to answer the embodied question with high confidence.

We demonstrate that given our real-time multimodal memory and hierarchical planning approach, the agent is able to accomplish long-horizon tasks with significantly fewer VLM planning steps, explores explainable task-relevant frontiers, and succeeds in EQA tasks with a higher rate than previous approaches. We demonstrate our results on the HM-EQA (165) and OpenEQA (175) datasets in the Habitat simulation environment (176) and also in the real world using the Hello Robot Stretch mobile manipulator in two home and two office scenes. The key contributions of this

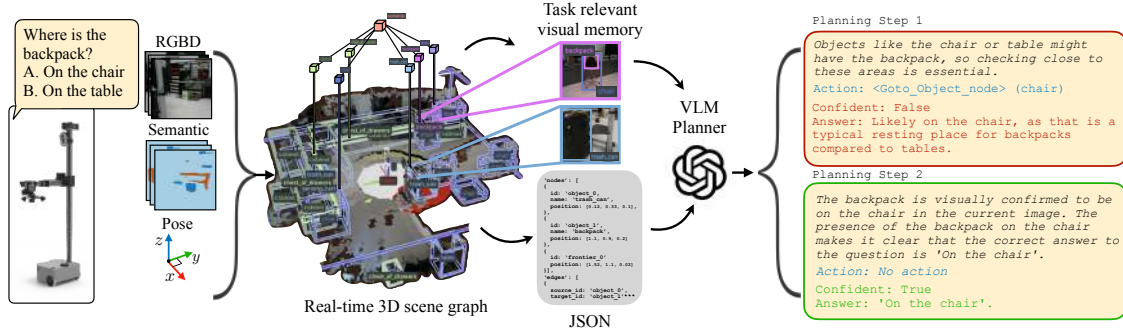


Figure 6.1: **Overview of GraphEQA:** A novel approach for utilizing real-time 3D metric-semantic hierarchical scene graphs and task-relevant images as multimodal memory for grounding vision-language based planners to solve embodied question answering tasks in unseen environments.

work are as follows:

- We present GraphEQA, a novel approach for using real-time 3D metric-semantic hierarchical scene graphs and task-relevant images as multimodal memory, for grounding VLMs to solve EQA tasks in unseen environments.
- We propose an approach to enrich 3DSGs with 1) semantically enriched frontiers and 2) semantic room labels.
- We propose a hierarchical VLM-based planning approach that exploits the hierarchical nature of the enriched 3DSG for structured exploration and planning.
- We provide extensive experimental results, both in simulation on the HM-EQA (165) and OpenEQA (175) datasets, and in the real-world in two home and two office environments, using the Hello Robot Stretch RE2 mobile manipulator.

6.2 Preliminaries

6.2.1 Hierarchical 3D Metric-semantic Scene Graphs

3D metric-semantic scene graphs (3DSGs) provide a structured, layered representation of environments and encode spatial, semantic, and relational information (177; 178; 179). Recent works like Clio (180), Hydra (174), and Open Scene Graphs (181) introduce efficient real-time frameworks for incremental construction of metric and semantic SG layers, encoding low-level geometries as well as high-level semantics at multiple levels of abstraction including objects, regions, rooms, buildings, etc. 3DSGs are comprised of layers that govern how buildings, rooms, objects, and regions are connected, a structure we leverage in GraphEQA.

Hydra 3DSGs are comprised of the following layers: *Layer 1 (bottom)*: a metric-semantic 3D mesh, *Layer 2*: objects and agent, *Layer 3*: regions or places, *Layer 4*: rooms, and *Layer 5 (top)*: building. Intra-layer edges between nodes denote

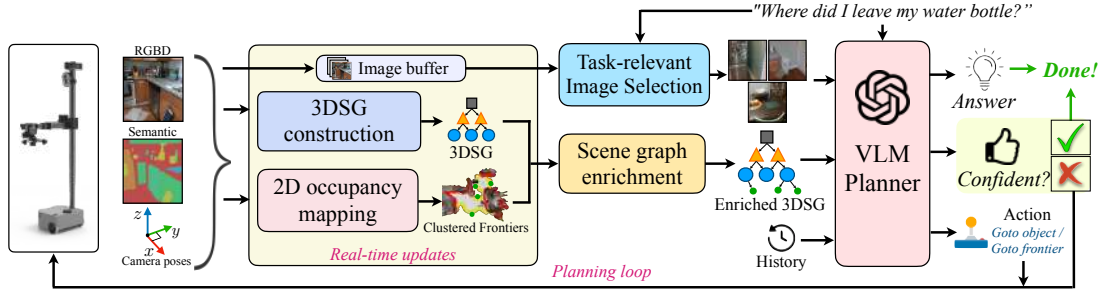


Figure 6.2: **Overall GraphEQA Architecture.** As the agent explores the environment, it used its sensor data (RGBD images, semantic map, camera poses and intrinsics) to construct a 3D metric-semantic hierarchical scene graph (3DSG) as well as a 2D occupancy map for frontier selection in real time. The constructed 3DSG is enriched as discussed in Section 6.3.2. From the set of images collected during each trajectory execution, a task-relevant subset is selected, called the task-relevant visual memory (Section 6.3.3). A VLM-based planner (Section 6.3.4) takes as input the enriched scene graph, task-relevant visual memory, a history of past states and actions, and the embodied question and outputs the answer, its confidence in the selected answer, and the next step it needs to take in the environment. If the VLM agent is confident in its answer, the episode is terminated, else the proposed action is executed in the environment and the process repeats.

'traversability', while inter-layer edges denote 'belonging'. For example, an edge between regions in Layer 3 denotes traversability between these regions. An edge between an object and a room means the object is located in that room. 3DSGs are constructed using RGB and depth images from the robot's camera, camera pose and camera intrinsics. Using off-the-shelf image segmentation models (182), the object nodes are assigned semantic object labels.

6.2.2 2D Occupancy Mapping and Frontier Detection

3D voxel-based occupancy maps are a simple and effective way for storing explored, occupied and unexplored regions of an environment for planning and navigation. As the robot explores, using depth data and camera intrinsics, occupancy of the voxels is updated using Volumetric Truncated Signed Distance Function (TSDF) fusion. TSDF integrates depth observations to update voxels as occupied or free, while areas beyond a certain threshold are marked unexplored. Typically, the 3D occupancy map is projected into 2D, where frontiers/boundaries between explored and unexplored regions are identified to guide further exploration. We employ this approach in our method for identifying frontiers, clustering them and adding them to the scene graph.

6.3 Method

6.3.1 Problem Formulation

An overview of our proposed method is shown in Figure 6.2. Given a multiple choice question q , we wish to find the correct answer $a^* \in \mathcal{A}$ where \mathcal{A} is the set of multiple-choice answers to question q available to the agent. To find a^* , the agent is equipped with a VLM-based planner $\mathcal{V}(q, \mathcal{S}_t^e, \{\mathcal{I}_k\}_{k=1}^K, \mathcal{H}_t, X_t) = (a_t, c_t, u_t)$, where \mathcal{S}_t^e is the enriched scene graph constructed online at planning time t , and includes frontier nodes from 2D the occupancy map (Section 6.2.2). $\{\mathcal{I}_k\}_{k=1}^K$ is a small set of task-relevant images maintained in memory (Section 6.3.3), a_t is the current best answer to the multiple choice question q and u_t is the next action the agent should take in the environment. \mathcal{H}_t represents the accrued history up to time t and X_t represents the current state of the agent. We query the VLM planner \mathcal{V} (Section 6.3.4) at time t with the inputs described above, with the scene graph \mathcal{S}_t^e continually being constructed and a set of images $\{\mathcal{I}_k\}_{k=1}^K$ chosen based on task and semantic relevance. The planner then outputs a high-level action u_t which is executed in the environment while the scene graph, visual memory, and frontiers are all updated in real time. In the following sections we provide details for each of these components.

6.3.2 Scene Graph Construction and Enrichment

We use Hydra (174) to construct a layered metric-semantic scene graph (see Appendix E.3), while also maintaining a 2D occupancy map of the environment depicting the explored, occupied, and unexplored navigable regions of the environment as mentioned in Section 6.2.2. We perform room and frontier enrichment steps to enable semantic-guided exploration and hierarchical planning.

Room enrichment: Room nodes in Hydra’s 3DSG are assigned generic labels such as R0, R1, etc. To enrich them with semantic information, we prompt an LLM to assign semantic labels to each of the room nodes. We use a simple prompt “Which room are these objects <object list> most likely located in?” where <object list> is the list of all objects located in a certain room in the scene graph. The output of the LLM is then used to update the room labels.

Frontier enrichment: To enrich our 3DSG with semantic information that can enable task-relevant exploration, we extract frontier nodes from the 2D occupancy map, cluster them, and add them as independent nodes to the scene graph. Next, we find top- j object nodes nearest to each clustered frontier node, within a maximum distance d . We add edges to the scene graph connecting each frontier node to its top- j object neighbors. This semantic information can now be utilized by a VLM-based planner to select the most semantically-relevant frontier to explore next. For general exploration, for example, it could be useful to choose frontier nodes near doors. We use $j = 3$ and $d = 2$ meters in our experiments, but can be varied based on the

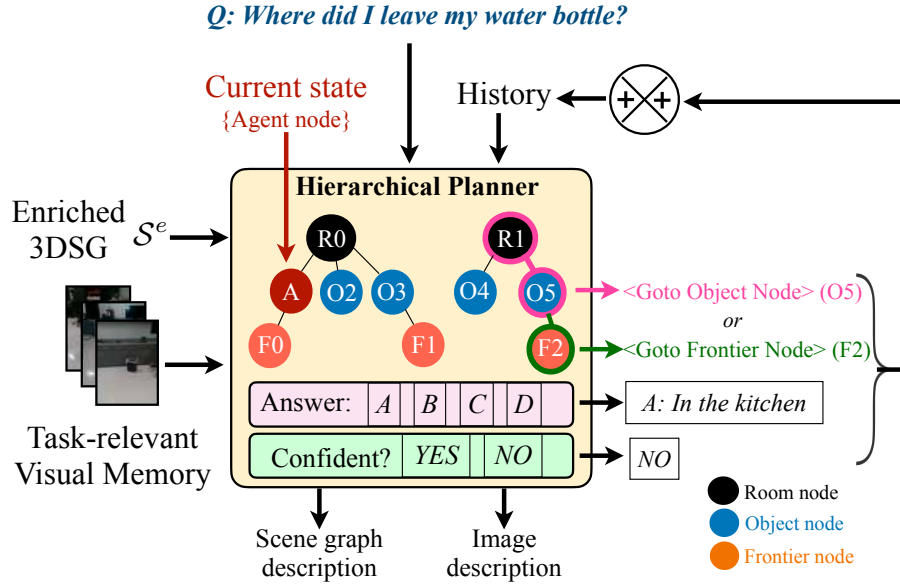


Figure 6.3: **VLM Planner Architecture.** The Hierarchical Vision-Language planner takes as input the question, enriched scene graph, task-relevant visual memory, current state of the robot (position and room) and a history of past states, actions, answers and confidence values. The planner chooses the next `<Goto_Object_node>` action hierarchically by first selecting the room node and then the object node. The `<Goto_Frontier_node>` action is chosen based on the object nodes connected to the frontier via edges in the scene graph. The planner is asked to output a brief reasoning behind choosing each action, an answer, confidence in its answer, reasoning behind the answer and confidence, the next action, a brief description of the scene graph, and the visual memory.

environment.

6.3.3 Task-relevant Visual Memory

During action-execution, images are stored in a buffer at a specified sampling frequency to avoid multiple similar repeated images. Images from this buffer, along with keywords from the question/task, are then processed using SigLIP (183) to obtain the text-image relevancy score for each image. Using this score, only the top- K most relevant images are maintained in the buffer and the rest are discarded. We use $K = 2$; we append these K images together, along with the agent’s current view, and use it as the visual input to the VLM planner at the next planning step, as shown in Figure 6.2.

6.3.4 Hierarchical Vision-Language Planner

Inputs: At every planning step t , the VLM planner takes as input a multiple-choice question q , the set of multiple-choice answers \mathcal{A} , the enriched scene graph \mathcal{S}_t^e , and the

task-relevant visual memory $\{\mathcal{I}_k\}_{k=1}^K$; K is the number of images in memory. Additionally, we provide the planner with a structured history \mathcal{H}_t and the agent’s current state X_t ; X_t is defined in the following format: "The agent is currently at node `<agent_node_id>` at position `<agent_position>` in room `<room_id>` `<room_name>`", where information in ‘<.’ is populated from \mathcal{S}_t^e (see Fig. 6.3).

Outputs: Given the above inputs, the planner must output an answer $a_t \in \mathcal{A}$ for q , a boolean value $c_t \in \{\text{True}, \text{False}\}$ representing whether it is confident in answering the question, its current numeric confidence level $p_t^c \in [0, 1]$, and the next action u_t that the agent should take. We require the planner to also output the reasoning behind the choice of action and its confidence in the rationale. Finally, the planner is required to plan the next few steps, selecting from two high-level action types: `<Goto_Object_node>(object_id)` and `<Goto_Frontier_node>(frontier_id)`, where `object_id` and `frontier_id` are selected from \mathcal{S}_t^e . Selecting an object node enables further visual examination of relevant visited areas. Selecting a frontier node enables visitation of unexplored areas. Finally, the planner is required to output a brief description of the scene graph as well as a brief description of the input images. We update the history \mathcal{H}_t such that $\mathcal{H}_{t+1} = X_t + a_t + c_t + p_t^c + u_t + \mathcal{H}_t$.

Hierarchical planner and frontier exploration: For `<Goto_Object_node>(object_id)` action types, we enforce a hierarchical planning behavior by requiring the planner to first reason about which room to go to by selecting a room node, then a region node (within the selected room), and finally the object node to go to. This planning behavior reflects the hierarchical structure of the 3DSG and forces the planner to reason about the hierarchical semantics of the scene to explore and answer the questions. For `<Goto_Frontier_node>(frontier_id)` action types, we require the planner to provide rationale for its choice of frontier node by referring to the object nodes connected to the selected frontier by edges in the scene graph, enforcing semantic reasoning in the frontier-selection process so that chosen frontiers are task-relevant and for information-gathering.

Termination condition: A planning episode is terminated when the planner outputs $c_t = \text{True}$ or $p_t^c > 0.9$, i.e., when it is confident in answering the question. The episode is also terminated if $t > T_{max}$, when the maximum allowed planning steps have been reached.

Prompt description: We provide the planner with a system prompt detailing how to understand the scene graph structure, and explain the criteria behind choosing the actions—hierarchically for object nodes and task-relevant or information-gathering for frontier nodes. We explain that the 3DSG can be imperfect/incomplete and that the planner should always seek visual confirmation before answering the question with confidence while employing the scene graph as a semantic map for examining

and exploring the scene. Finally, we prompt the VLM to provide a brief description of the scene graph and the input images, focusing on elements in the scene that are relevant to the current task. The complete prompt is available in Appendix E.4.

6.4 Experimental Setup

We identify the key research questions that this work aims to evaluate: **Q1)** Do hierarchical 3D scene graphs provide an effective **metric-semantic memory** for solving **embodied question answering** tasks? **Q2)** How does the **real-time** nature of GraphEQA compare to offline approaches that provide the planner with full-state scene graphs? Specifically, we aim to evaluate if GraphEQA can utilize incrementally constructed state information to solve EQA tasks and terminate based on **confidence**, without needing to acquire full state information. **Q3)** Can GraphEQA effectively combine and reason about the **high-level, semantically-sparse and task-agnostic** information offered by **scene graphs** and the **local, semantically-rich and task-relevant** information from **visual memory** to actively take information gathering actions until it can confidently answer an embodied question?

6.4.1 Baselines and Ablations

To evaluate our method and answer the above research questions, we compare against several baselines and focus on methods that employ VLM-based planners for solving EQA or object goal navigation tasks. We compare against a strong baseline, **Explore-EQA** (165), which calibrates Prismatic-VLM (184) to answer embodied questions confidently while maintaining a 2D semantic memory and using prompted images to guide exploration. Note that ExploreEQA is always executed for a pre-specified maximum number of steps, with the highest confidence step chosen to answer the question, while GraphEQA terminates based on a confidence threshold. We implement additional variants of ExploreEQA with newer foundation models—e.g., GPT4o, Llama 4 Maverick, and Gemini 2.5 Pro—for fair comparison with respective VLM variants of GraphEQA.

We also compare GraphEQA against a modified version of **SayPlan** (166) which we call **SayPlanVision**. Similar to SayPlan, SayPlanVision first constructs a scene graph of the whole scene offline and then uses this scene graph for planning. For fair comparison, we further augment SayPlanVision with some abilities of GraphEQA in order to evaluate the effectiveness of our real-time approach and to answer **Q2**; we provide it with a task-relevant visual memory and confidence-based termination.

We further evaluate our method for **Q3**, using two ablations: **GraphEQA-SG**, where the planner only has access to the real-time 3DSG and does not have access to images; and **GraphEQA-Vis**, where the planner only takes the visual memory as input and exploration is done via random frontier-based exploration. These ablations

will help us evaluate the benefits of multimodality in GraphEQA.

Experimental Settings: Since we focus on multi-room environments, we evaluate GraphEQA and the baselines mentioned in Section 6.4.1 in simulation in HabitatSim (185) on scenes from HM3D-Semantics (176) on the HM-EQA and OpenEQA datasets and in the real-world in two unique home environments. We perform numerous experiments with different foundation models as the VLM planner, including GPT4o, Gemini 2.5 Pro, and Llama 4 Maverick. For the real-world setup, we deploy and evaluate our approach on the *Hello Robot Stretch RE-2*¹ mobile manipulation platform. All experiments are conducted on a desktop machine with two (2) NVIDIA TITAN RTX GPUs, 64GB of RAM, and an Intel i9-10900K CPU.

Resource Allocation: The average token count is broken down as 475 per image (3 images), 1133 for the prompt, and on average 5,425 scene graph tokens, for a total 7983 tokens per VLM step.

Metrics: We use the following three metrics to compare against the baselines and ablations in Section 6.4.1: 1) *Success Rate* (%): an episode is considered a success if the agent answers the embodied question correctly with high confidence; 2) *Average # Planning Steps*: For successful episodes, we calculate the average number of VLM planning steps. Note that while Explore-EQA runs for a preset maximum number of steps and post-calculates the maximum confidence step, we report the number of steps taken *until* the max confidence step; 3) *Average Trajectory Length* (meters): for successful episodes, we calculate the average length of the path traveled by the robot. For SayPlanVision, this includes the path traveled to generate the full scene graph.

6.5 Experimental Results

Comparison to Baselines. Table 6.1 shows simulation results comparing **GraphEQA** to the baselines discussed in Section 6.4.1 on the HM-EQA and OpenEQA datasets. Overall, GraphEQA outperforms all other baselines. Compared to **Explore-EQA**, our method completes tasks in significantly fewer planning steps and with lower trajectory length, indicating more efficient navigation. We also observe that the GPT and Llama variants of Explore-EQA have lower success rates than **Explore-EQA**, with qualitative results indicating overconfidence in VLMs’ predictions (see Appendix E.8), leading to terminating episodes before exploring sufficiently. We note that Explore-EQA’s Gemini variant performs better than **Explore-EQA**, likely due to the inherent spatial reasoning capabilities of Gemini 2.5 Pro. GraphEQA outper-

¹GraphEQA leverages the open-source codebase for Hello Robot’s stretch.ai at <https://github.com/hello-robot/stretch.ai>

Table 6.1: Comparison to simulation baselines for HM-EQA and OpenEQA datasets: Success rate (%), average # of planning steps over successful trials, and L_τ average trajectory length over successful trials. Methods with a [†] indicate our implementations of that particular baseline.

Method	HM-EQA			OpenEQA		
	Success Rate (%) \uparrow	#Planning Steps \downarrow	L_τ (m) \downarrow	Success Rate (%) \uparrow	#Planning Steps \downarrow	L_τ (m) \downarrow
Explore-EQA (165)	51.7	18.7	38.1	55.3	20.8	39.7
Explore-EQA-GPT4o [†]	46.4	3.4	6.3	46.4	4.88	8.30
Explore-EQA-Llama4-Mav [†]	43.8	5.58	10.4	48.1	4.25	7.60
Explore-EQA-Gemini-2.5Pro [†]	54.3	6.40	12.3	53.0	6.45	10.9
SayPlanVision [†]	54.8	2.6	5.3	-	-	-
GraphEQA-GPT4o	63.5	5.1	12.6	69.1	3.97	8.29
GraphEQA-Llama4-Mav	57.7	2.36	3.59	53.3	2.37	3.45
GraphEQA-Gemini-2.5Pro	67.0	2.94	7.41	62.0	2.16	4.03

forms **SayPlanVision** even though SayPlanVision has access to the complete scene graph. We discuss these results in more detail below.

Baseline And Ablation Study: Regarding **Q1**, we observe from Table 6.1 that **GraphEQA** has higher success rate, compared to all **Explore-EQA** variants, across both HM-EQA and OpenEQA datasets, without the need to build an explicit 2D semantic task-specific memory. This demonstrates the capability of 3DSGs to provide an effective metric-semantic memory for EQA tasks. We also observe that GraphEQA requires a significantly lower number of planning steps as compared to **Explore-EQA**. This is because, unlike **Explore-EQA**, GraphEQA does not entirely rely on images as input to the VLM planner for building the semantic memory as well as planning, as this would constrain the planner to choose from only regions that are visible in the current image. **GraphEQA**, on the other hand, can use the hierarchical structure of the scene graph as well as semantically-enriched frontier nodes to plan across the entire explored space.

Additional error analysis in Appendix E.8 reveals that the GPT variant of Explore-EQA has a significantly high percentage of *false positives*, i.e., questions that are answered successfully using commonsense reasoning/guessing, but without grounding the answer in the current scene. This provides additional evidence of the effectiveness of 3DSGs in enabling *semantic exploration* by grounding the planner in the current environment. Qualitatively, we observe that actions chosen by the planner, `<Goto_Object_node>(object_id)` and `<Goto_Frontier_node>(frontier_id)`, are task-specific and explainable. For more qualitative results please refer to Appendix E.12. We limit this error analysis to only the GPT variant since it is performed manually by a human.

For **Q2**, we observe from Table 6.1 that GraphEQA performs better than **SayPlanVision** which has access to the complete scene graph. This is a surprising result since it is expected that, given full scene graph information, SayPlanVision would outperform GraphEQA across all metrics. However, from a qualitative analysis of the results for **SayPlanVision**, we observe that given access to the complete scene

graph, the context is too large, providing for a much harder inference problem for the VLM. As a result, the agent is overconfident about its choice of object node actions, leading to shorter trajectory lengths in successful cases, but also to increased failure cases. This points to an interesting advantage of our real-time approach—that *incrementally* building memory by exploring task-relevant regions and maintaining a more concise representation benefits EQA tasks.

For **Q3**, the ablation results in Table 6.2 for **GraphEQA-SG** show lower success rate, higher average planning steps, and higher average trajectory length, compared to GraphEQA, as it only uses the 3DSG as textual input to the VLM planner, demonstrating that a semantic scene graph constructed using closed-set semantics and without any task-specific semantic enrichment will provide an incomplete and insufficient environment understanding; GraphEQA’s task-relevant visual memory and task-specific enrichment are crucial for solving EQA tasks. Furthermore, we note that the performance of the vision-only ablation **GraphEQA-Vis** *also* suffers: this is because the agent takes random exploratory actions in the environment, with no semantic understanding of the scene structure to guide exploration. However, qualitatively we observe that without access to a scene graph to ground the agent in the current environment, GraphEQA-Vis exhibits overconfidence (taking very few planning steps) and tries to answer the question solely based on the current image. **GraphEQA** outperforms all ablations,

providing clear evidence on the utility of a multimodal approach that combines global spatial and semantic information from 3D scene graphs with local but rich semantic information from images, for solving challenging EQA tasks. We also observe from Table 6.1 that GraphEQA exhibits lower average planning steps and lower average trajectory length, while retaining a higher success rate, highlighting the capability of GraphEQA to reason about multimodal data in the form of semantically sparse and task-agnostic scene graphs and more semantically rich information in the form of task-relevant visual memory. Additional ablations are available in Appendix E.6.

Table 6.2: Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.

Ablation	Succ. Rate (%)	#Planning steps	L_τ (m)
GraphEQA-SG	13.6	8.8	33.0
GraphEQA-Vis	45.7	1.0	3.9
GraphEQA	63.5	5.1	12.6

Real-world Experiments: We deploy GraphEQA on the **Hello Robot Stretch RE2** platform across two home and two office environments. We conduct ten trials in Home (a) and five trials in Home (b), as summarized in Figure 6.4, which illustrates representative runs from each setting. For each environment, we design a custom set of Embodied Question Answering (EQA) tasks (see Appendix E.10), aligned with the task categories described in Appendix E.2. To construct the underlying 3D metric-semantic scene graph, GraphEQA employs Detic (182) for semantic segmentation and integrates RGB-D images along with camera intrinsics and extrinsics as input to the



Figure 6.4: **Images from real-world experiments**, deploying GraphEQA on the Hello Robot Stretch RE2 platform in two unique home environments (a, b). Each set of images is from the head camera on the Stretch robot, representing the top-K task-relevant images at each planning step as it constructs the scene graph and attempts to answer the question with high confidence. Provided under the images are planning step, answers, confidence, and explanations output from the VLM planner.

Hydra scene graph generator (174). Results for the office environments are provided in Appendix E.11.

6.6 Conclusions

We present GraphEQA, an approach for solving embodied question answering tasks in unseen environments by grounding a vision-language based planner in the context of the current environment by providing as input textual representations of real-time 3D metric-semantic scene graphs and a task-relevant visual memory. We show that GraphEQA achieves improved performance over existing approaches on EQA tasks in both the HM-EQA and OpenEQA benchmarks, and validate performance across both *closed* and *open* multimodal foundation models. Furthermore, we validate its practical applicability through real-world deployments in two home and two office environments.

6.7 Limitations

A limitation of this approach is reliance on off-the-shelf segmentation and detection models for creating semantic maps required for 3DSG construction. The performance of our approach, hence, is directly impacted by the performance of the detection method used and the semantic categories in the scene graph are limited to the categories detected by the segmentation model. An exciting direction for future work includes enriching the scene graph online with task-relevant node and edge features using open-set models. Another limitation of our approach is that VLM-based

planners can be overconfident or underconfident when answering embodied questions. Grounding calibrated models in real-time multi-modal memory is another important direction for future work. Additionally, GraphEQA is currently limited to single-floor scenes, and is unable to traverse multiple floors or buildings. Accommodating multi-floor scenes is within the scope of future work.

7

CONCLUSIONS

This thesis explores the challenge of enabling robots to perform dynamic manipulation tasks that require both generalization and reactivity. Unlike quasi-static settings, dynamic manipulation involves long-term reasoning about object interactions while also adapting to unexpected contacts and disturbances in real time. To address this, we presented a series of contributions that collectively advance toward generalist, safe, and lifelong robotic agents. These contributions span reactive and predictive control, structured dynamics with attention, multi-resolution sensing with VLM-based semantic understanding, safety-aware policies, and lifelong planning and memory for open-world environments.

Reactive and Predictive Differentiable Control for Switching Dynamical Systems: This work shows that dynamic manipulation benefits from policies that couple *foresight* with *reaction* while explicitly accounting for contact-rich, switching dynamics. In (1), we learn the parameters of a differentiable LQR policy that (i) predicts future interactions to shape control before they occur, and (ii) reactive to high-frequency force-torque feedback to remain reactive to unanticipated contacts and modeling errors. Crucially, by learning the cost structure in a single scenario, the same policy generalizes to new goal conditions simply by updating the goal terms thus preserving the overall behavior style without retraining. Together, these elements deliver a contact-aware controller that is both predictive and robustly reactive, enabling reliable performance across dynamic tasks and novel goals.

Learning Structured Dynamics for Generalization: Building on this foundation, we extend our approach to more complex, switching dynamical systems where interactions involve multiple objects and non-linear dynamics. Here, we emphasize how structure and attention enable scalable generalization. In (2), we jointly learn a dynamic graph structure and a locally linear dynamics model. This structured representation enables the policy to attend to relevant objects and interactions while ignoring irrelevant distractors, which in turn improves generalization. As a result, the learned models transfer to novel objects, unseen interaction patterns, and increasingly cluttered environments.

Generalization and Reactivity via Multiresolution sensing: We then turn to

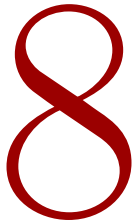
the challenge of deploying these policies in open-world environments where semantics, object properties, and interactions are diverse. This requires unifying semantic generalization with reactive control. Prior works typically emphasize one but neglect the other, often focusing on quasi-static manipulation. We achieve both by leveraging multi-resolution sensing (3): visual and language modalities provide semantic understanding, language-conditioning and rough localization, while force–torque and proprioceptive feedback ensure reactivity and dexterity during contact-rich manipulation. Transformer-based architectures implicitly attend to the most relevant modalities depending on context. Zero-shot generalization to new semantic variations is achieved by incorporating frozen pretrained vision-language models (VLMs), while fine-tuned smaller policies integrate this semantic reasoning into closed-loop control.

Safety in Dynamic Interactive Tasks: Safety is a critical requirement when robots operate around diverse objects, including fragile items and humans, where naive reactivity or collision avoidance is insufficient. For dynamic tasks, robots must reason about long-term safe interactions while remaining agile and efficient. We address this, in VLTSafe (5), by formulating a reach-avoid reinforcement learning problem to learn policies that balance task success with long-term safety guarantees. Transformer-based object-centric representations allow generalization to varying numbers of objects at test time, while pretrained VLMs provide semantic grounding for safety reasoning with novel object categories. This combination ensures both physical safety and semantic robustness in real-world deployments.

Towards Lifelong Robotic Operation: Finally, this thesis takes a step towards lifelong robotic operation in open-world environments. We introduce GraphEQA (4), a system for long-horizon exploration and planning in unseen environments. GraphEQA grounds a VLM-based planner in a real-time metric-semantic multimodal memory that captures both task-agnostic structure and task-relevant details. By maintaining structured memory and using hierarchical planning, robots achieve improved task efficiency, robustness, and generalization across large-scale, previously unseen environments.

Together, this work demonstrates that dynamic, generalist robotic manipulation relies on attention and structure to prioritize the most relevant sensing modalities, objects, and interactions, while at the same time maintaining both reactivity and foresight to predict future interactions, ensure safety, and adapt to unanticipated contacts. As robotics increasingly embraces large-scale data-driven learning, this thesis underscores the continued importance of modularity, structure, hierarchy, and inductive biases. These principles not only enable agility, dexterity and reliability, but also explainability, generalization, and data-efficient learning. This work moves us closer to generalist embodied agents that combine the semantic reasoning capabilities of foundation models with the reactivity and safety of robust policies, laying the foundation for robots that can operate safely and autonomously in dynamic, open-world environments. By enabling such capabilities, these advances have the potential

to impact real-world domains ranging from household and industrial automation to healthcare and human-robot collaboration, ultimately bringing us closer to robots that can assist, adapt, and interact effectively in diverse human-centered settings.



DIRECTIONS FOR FUTURE WORK

8.1 Lifelong Robotic Operation Through Structured Learning and Adaptation

To enable robotic systems to support humans across diverse settings – ranging from caregiving roles to hazardous environments – they must be explainable, reliable, and adaptable. This thesis takes steps toward that goal by emphasizing modularity, structure, and hierarchy in system design. These principles serve as foundational components for building generalizable and interpretable robot behavior. However, a significant gap remains in closing the loop between key hierarchical modules: memory, high-level planning and reasoning, and low-level execution. Achieving true lifelong learning requires *sample-efficient* mechanisms that integrate continuous feedback – whether from human intervention or from observing task failures – and use that feedback to adapt dynamically across both high-level and low-level layers of the system.

Critical directions for future work include advancing techniques for reasoning about success and failure, enabling online adaptation of both control policies and planning strategies, and developing methods to learn effectively from sub-optimal or failed trajectories. Moreover, recent efforts have demonstrated the feasibility of learning from a single or very few human demonstrations (186; 187) by leveraging tools such as hand-tracking and object-centric trajectory understanding. Extracting *task-relevant* structure from such demonstrations and generalizing it to semantically similar tasks promotes sample-efficient learning and improves transfer across related domains. Recent vision-language models (VLMs), which encode a common-sense understanding of objects and tasks, offer a promising avenue for such generalization. Furthermore, such models can facilitate not only the updating of low-level skills and high-level plans but also the refinement of memory modules – specifically, what information to attend to based on new experiences. A structured planner can then use this evolving memory and knowledge to transfer skills across novel but semantically related objects or scenarios. Ultimately, enabling such continuous, structured adaptation is key to achieving robust lifelong robot operation in the real world.

8.2 Open-World Safety through Uncertainty-Aware and Human-in-the-Loop Control

When deploying robots in environments shared with humans, safety becomes the foremost priority. In this thesis, we explored learning parameterized policies that respect specific semantic constraints, particularly in tabletop manipulation tasks. While these controlled settings offer a starting point, real-world applications – especially in caregiving and hospital environments – demand far more comprehensive safety considerations. These environments require reasoning about not just spatial constraints, but also how different objects should be handled safely based on their function, fragility, or medical importance, as well as how to interact appropriately around or with humans. Furthermore, these constraints should be grounded in multimodal perception – including vision, audio, and proprioception – since reliable ground-truth state estimation is often unavailable in real-world, dynamic environments.

A major challenge in scaling robotic systems to open-ended tasks is that greater generalization and multimodal learning often come at the cost of increased uncertainty and reduced reliability. While systems like VLTSafe demonstrate more safety-aware behavior, they remain fundamentally data-driven and lack formal safety guarantees. To address this, future systems must be structured in a way that allows external monitoring and human oversight. Robotic operators should be able to inspect and approve the robot’s intended actions, providing corrective feedback (188) when necessary. This human-in-the-loop supervision not only improves immediate safety but also provides valuable signals for the robot to learn and refine its policy over time.

Another critical direction is to design robots with explicit uncertainty estimation capabilities. Techniques such as Bayesian deep learning, ensemble models, and conformal prediction (189) can help quantify confidence in action decisions. Robots equipped with such models should be able to assess whether their planned actions satisfy safety constraints with high confidence. If uncertainty is too high, they must defer to human feedback, seek additional sensory input, or abstain from acting altogether. Systems like SafeDagger (190) begin to explore these ideas by combining introspective reasoning and fallback strategies during unsafe or ambiguous states. Developing this kind of introspective and conservatively adaptive behavior is essential for building trustworthy systems capable of operating safely in dynamic, human-centered environments.

I

APPENDICES

A

APPENDIX FOR CHAPTER 2

A.1 Training details

The classifier, for all the experiments, is a single layer neural network with 8 hidden nodes. The output layer has nodes equal to the number of modes and softmax activation. The other hyperparameters used during training (system identification as well as learning control parameters) are as follows:

Table A.1: Hyperparameters

Optimizer	ADAM (61)
Learning rate	0.01
β_1	0.9
β_2	0.999
ϵ	1e-04
Length of each trajectory	12.5sec
dt	0.05sec
batch size	10

A.2 Task 1: 1-object pickup

This task involves a gripper of mass m^g picking up an object of mass m^o while moving in a 2D plane. The independent dynamics of the gripper and object can be

written as,

$$\begin{aligned} \mathbf{x}_{t+1}^g &= \mathbf{A}_t \mathbf{x}_t^g + \mathbf{B}_t^g \mathbf{u}_t^g, \quad \mathbf{x}_{t+1}^o = \mathbf{A}_t \mathbf{x}_t^o + \mathbf{B}_t^o \mathbf{u}_t^o, \text{ where} \\ \mathbf{x}_t^g &= \begin{bmatrix} q_{x_t}^g \\ q_{y_t}^g \\ \dot{q}_{x_t}^g \\ \dot{q}_{y_t}^g \end{bmatrix}, \mathbf{A}_t = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B}_t^g = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt/m^g & 0 \\ 0 & dt/m^g \end{bmatrix} \\ \mathbf{x}_t^o &= \begin{bmatrix} q_{x_t}^o \\ q_{y_t}^o \\ \dot{q}_{x_t}^o \\ \dot{q}_{y_t}^o \end{bmatrix}, \mathbf{B}_t^o = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt/m^o & 0 \\ 0 & dt/m^o \end{bmatrix}, \mathbf{u}_t^g = \begin{bmatrix} u_{x_t}^g \\ u_{y_t}^g \end{bmatrix}, \mathbf{u}_t^o = \begin{bmatrix} u_{x_t}^o \\ u_{y_t}^o \end{bmatrix} \end{aligned}$$

q_{x_t}, q_{y_t} and $\dot{q}_{x_t}, \dot{q}_{y_t}$ are positions and velocities respectively. It is important to note here that the object is unactuated ($u_{x_t}^o = 0$ at all time t) but when grasped is affected by the force applied by the gripper. We consider two modes: Mode 1 is when gripper is free and mode 2 is when the gripper has grasped the object and is moving with it. The combined dynamics of gripper and object in different modes $\rho_t \in \{1, 2\}$ is as given in (2.1), where

$$\begin{aligned} \mathbf{x}_t &= \begin{bmatrix} \mathbf{x}_t^g \\ \mathbf{x}_t^o \end{bmatrix}, \mathbf{A}_{1_t} = \mathbf{A}_{2_t} = \begin{bmatrix} \mathbf{A}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_t \end{bmatrix} \\ \mathbf{B}_{1_t} &= \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{0} \end{bmatrix}, \mathbf{B}_{2_t} = \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{B}_t^o \end{bmatrix}, \mathbf{u}_t = \begin{bmatrix} u_{x_t}^g \\ u_{y_t}^g \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{C}_{\rho_{t-1}, \rho_t} &= \boldsymbol{\rho}_{t-1}^\top \boldsymbol{\rho}_t \mathbf{I} + (1 - \boldsymbol{\rho}_{t-1}^\top \boldsymbol{\rho}_t) \mathbf{C} \\ C_{1,2} = C_{2,1} &= \frac{1}{m^g + m^o} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m^g & 0 & 0 & 0 & m^o & 0 \\ 0 & 0 & 0 & m^g & 0 & 0 & 0 & m^o \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & m^g & 0 & 0 & 0 & m^o & 0 \\ 0 & 0 & 0 & m^g & 0 & 0 & 0 & m^o \end{bmatrix} \end{aligned}$$

Here, we observe that there are only two parameters that we need to learn to populate the above matrix, $C_1 = \frac{m^g}{m^g + m^o}$ and $C_2 = \frac{m^o}{m^g + m^o}$. Thus, during system identification we learn the four parameters: C_1, C_2, m^g, m^o in addition to the classifier.

Expert trajectories for this task are generated using the following cost parameters for the respective modes in iLQR,

$$Q_1 = Q_2 = \text{diag}([1, 1, 1, 1, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}])$$

$$R_1 = \text{diag}([1, 1]), R_2 = \text{diag}([0.5, 0.5])$$

where $\text{diag}()$ represents a diagonal matrix. During training we assume that the cost matrices diagonal.

A.2.1 Task 2: 2-object pickup

This task considers 3 modes: Mode 1 is when the gripper is free, mode 2 is when the gripper has grasped object 1 and mode 3 is when gripper has grasped object 2. The combined dynamics (2.1) can be written using:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{x}_t^g \\ \mathbf{x}_t^{o1} \\ \mathbf{x}_t^{o2} \end{bmatrix}, \mathbf{A}_{1_t} = \mathbf{A}_{2_t} = \mathbf{A}_{3_t} = \begin{bmatrix} \mathbf{A}_t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_t \end{bmatrix}$$

$$\mathbf{B}_{1_t} = \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \mathbf{B}_{2_t} = \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{B}_t^{o1} \\ \mathbf{0} \end{bmatrix}, \mathbf{B}_{3_t} = \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{B}_t^{o1} \\ \mathbf{B}_t^{o2} \end{bmatrix}, \mathbf{u}_t = \begin{bmatrix} u_{x_t}^g \\ u_{y_t}^g \end{bmatrix},$$

In the interest of space, we write the C matrix such that it maps the velocities just before collision to the velocities just after collision, thus ignoring the rows that identically map the position before collision to position after collision.

$$C_{1,2} = C_{2,1}$$

$$= \frac{1}{m^g + m^{o1}} \begin{bmatrix} m^g & 0 & m^{o1} & 0 & 0 & 0 \\ 0 & m^g & 0 & m^{o1} & 0 & 0 \\ m^g & 0 & m^{o1} & 0 & 0 & 0 \\ 0 & m^g & 0 & m^{o1} & 0 & 0 \\ 0 & 0 & 0 & 0 & m^g + m^{o1} & 0 \\ 0 & 0 & 0 & 0 & 0 & m^g + m^{o1} \end{bmatrix}$$

$$C_{2,3} = \frac{1}{m^g + m^{o1} + m^{o2}} \begin{bmatrix} m^g & 0 & m^{o1} & 0 & m^{o2} & 0 \\ 0 & m^g & 0 & m^{o1} & 0 & m^{o2} \\ m^g & 0 & m^{o1} & 0 & m^{o2} & 0 \\ 0 & m^g & 0 & m^{o1} & 0 & m^{o2} \\ m^g & 0 & m^{o1} & 0 & m^{o2} & 0 \\ 0 & m^g & 0 & m^{o1} & 0 & m^{o2} \end{bmatrix}$$

Thus, to populate each $C_{i,j}$ matrix we need 9 independent parameters. We learn these parameters along with the masses m^g, m^{o1}, m^{o2} and classifier during system identification. It is interesting to note here that we only observe $C_{1,2}$ and $C_{2,3}$ in our demonstrations for this task, hence even though we define all the other $C_{i,j}$ s as training variables they are never learned and do not affect the training time.

Expert trajectories for this task are generated using the following cost parameters for the respective modes in iLQR,

$$\begin{aligned} Q_1 &= Q_2 = Q_3 \\ &= \text{diag}([1, 1, 1, 1, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}]) \\ R_1 &= \text{diag}([1, 1]), R_2 = R_3 = \text{diag}([0.5, 0.5]) \end{aligned}$$

where $\text{diag}()$ represents a diagonal matrix. Here again during training we assume that the cost matrices diagonal.

A.2.2 Task 3: Pickup task with ground interactions

This task captures four modes: Mode 1 is when the gripper moves in free space, mode 2 is when the gripper comes in contact with the table and moves along it, mode 3 is when the gripper comes in contact with the object on the table and pushes it along the table and mode 4 is when the gripper picks up the object off the table and takes it to the goal location. The dynamics for each mode can be written as,

$$\begin{aligned} \mathbf{x}_t &= \begin{bmatrix} \mathbf{x}_t^g \\ \mathbf{x}_t^o \end{bmatrix}, \mathbf{A}_{1_t} = \mathbf{A}_{2_t} = \mathbf{A}_{3_t} = \mathbf{A}_{4_t} = \begin{bmatrix} \mathbf{A}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_t \end{bmatrix} \\ \mathbf{B}_{1_t} = \mathbf{B}_{2_t} &= \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{0} \end{bmatrix}, \mathbf{B}_{3_t} = \mathbf{B}_{4_t} = \begin{bmatrix} \mathbf{B}_t^g \\ \mathbf{B}_t^o \end{bmatrix}, \mathbf{u}_t = \begin{bmatrix} u_{x_t}^g \\ u_{y_t}^g \end{bmatrix}, \end{aligned}$$

Again, writing down the impact dynamics while ignoring the rows that identically map positions before collision to after collision,

$$\begin{aligned} C_{1,2} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ C_{2,3} = C_{3,4} &= \frac{1}{m^g + m^o} \begin{bmatrix} m^g & 0 & m^o & 0 \\ 0 & m^g & 0 & m^o \\ m^g & 0 & m^o & 0 \\ 0 & m^g & 0 & m^o \end{bmatrix} \end{aligned}$$

Thus, each of the above matrices can be populated using 4 independent parameters which we learn along with the masses m^g, m^o and classifier during system identification. The expert trajectories were generated using the following cost parameters for

each mode,

$$\begin{aligned}
 Q_1 &= \text{diag}([1, 10, 1, 1, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}]) \\
 Q_2 &= Q_3 = \text{diag}([1, 1, 1, 1, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}]), \\
 Q_4 &= \text{diag}([1, 1, 0.01, 0.01, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}]), \\
 R_1 &= R_2 = \text{diag}([1, 1]), R_3 = \text{diag}([0.5, 0.5]), R_4 = \text{diag}([0.1, 0.1])
 \end{aligned}$$

where $\text{diag}()$ represents a diagonal matrix. Here again during training we assume that the cost matrices diagonal.

B

APPENDIX FOR CHAPTER 3

B.1 Experiment Details

Box 2D data: The trajectory length for all the box2D tasks is 10 seconds with $dt = 0.05$. The simulation environment is setup such that when two objects come in contact they undergo a purely inelastic collision and a rigid link is formed between them, thus simulating grasping. We collect 40 trajectories for training and 10 for validation by sampling the initial position of the objects/doors, the gripper and the goal in a $2\text{m} \times 2\text{m}$ 2D space. The masses for the gripper and objects are 1 kg each. The batch size used for training is 64.

Issac Gym data: The trajectory length for the Isaac Gym Franka task is 4 seconds with $dt = 0.01$. We collect 200 trajectories for training and 20 for validation by sampling the initial position of the objects and the goal in a $0.4\text{m} \times 0.7\text{m} \times 0.3\text{m}$ 3D region of interest in front of the robot. The mass of the object for pickup is 0.4 kg. The batch size used for training is 256.

Real Franka data: The trajectory length for the Real world Franka task is 10 seconds with $dt = 0.001$. We collect 25 trajectories for training and 5 for validation by sampling the initial position of the objects and the goal in a $0.4\text{m} \times 0.7\text{m} \times 0.3\text{m}$ 3D region of interest in front of the robot. The mass of the object for pickup is 0.83 kg. The batch size used for training is 128.

B.2 Additional results

B.2.1 Learned dynamic parameters and edge predictions

For the Box2D tasks 1 & 2, the dynamics are linear and represent that of a point-mass system. We visualize the learned mass, stiffness and damping parameters and the predicted edge activations for a trajectory in Fig. ???. The gripper reaches the object and makes contact at around time step 50 as observed in the position plots as

well as in the edge activation plot (top left in Fig. ??). The edge between the gripper and the object is inactive before contact and active after contact. Before contact the mass of the gripper is predicted to be 1kg and the object mass of predicted to be 0kg. This is because since we apply the same control to all the nodes of the graph, actuated or unactuated, the object mass needs to be zero such that the effect of the actions on it is zero before contact. At the moment of contact we observe an impulse in the velocity plots wherein an inelastic collision happens between the gripper and the object. This is also reflected as a spike in the dynamic parameter plots and represents impact parameters. After contact and hence pickup, the masses of the gripper and the object are predicted to be 0.5kg each. This is because since the control effort is now applied to a total mass of 2kg, each node is effectively sharing half the control effort, which reflects as their individual masses being halved. The other parameters k_x, k_y, c_x and c_y are predicted to be zero (except at impact) as expected for a point mass system.

B.3 CONTROL USING LEARNED LOCALLY LINEAR MODELS

As mentioned in 3.4 each of the learned models are used for control using iLQR-MPC. The iLQR procedure is as follows:

Algorithm 3 Iterative LQR

(i) represents the iteration number
 T is the length of the trajectory
Given: Initial state \mathbf{x}_{init} .
Parameters: $\mathbf{A}_t, \mathbf{B}_t, \mathbf{x}_f, \mathbf{Q}_t, \mathbf{R}_t$
procedure iLQR(Parameters)
 while not converged **do**
 $\mathbf{x}_{0:T}^{(i)}, \mathbf{K}_{0:T-1} \leftarrow \text{LQR}(\mathbf{A}_t, \mathbf{B}_t, \mathbf{x}_f, \mathbf{Q}_t, \mathbf{R}_t)$
 converged $\leftarrow \|\mathbf{x}_{0:T}^{(i+1)} - \mathbf{x}_{0:T}^{(i)}\|_2 + \|\mathbf{u}_{0:T-1}^{(i+1)} - \mathbf{u}_{0:T-1}^{(i)}\|_2$ is small
 end while
end procedure

The MPC procedure is given in Algorithm 2. Note that the observed contact modes are used to evaluate a posterior over the edge activations and this posterior is used for future edge activation predictions.

Algorithm 4 Model Predictive Control

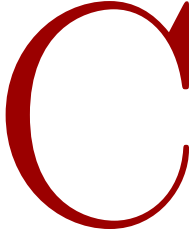
T_{LQR} is time horizon for execution before replanning
 $N_{MPC} = T/T_{LQR}$
 T is the length of the trajectory
 t is the current time step
 $t \leftarrow 0$
Given: Initial state \mathbf{x}_{init} , Initial edge activation \mathbf{p}_0^{ij} , Learned model \mathbf{T}
Parameters: $\mathbf{x}_f, \mathbf{Q}_t, \mathbf{R}_t$
for n in N_{mpc} **do**
 $\mathbf{A}_t, \mathbf{B}_t \leftarrow \mathbf{T}(\mathbf{p}_t^{ij}, \mathbf{x}_t)$
 $\mathbf{K}_{t:T} \leftarrow \text{iLQR}(\mathbf{x}_t, \mathbf{A}_t, \mathbf{B}_t, \mathbf{R}_t, \mathbf{Q}_t, \mathbf{x}_f)$
 for i in T_{LQR} **do**
 $\mathbf{a}_t \leftarrow \mathbf{K}_t(\mathbf{x}_t - \mathbf{x}_f)$
 $\mathbf{x}_{t+1} \leftarrow \text{step}(\mathbf{a}_t)$
 $\tilde{\mathbf{p}}_t^{ij} \leftarrow z\mathbf{p}_t^{ij}, \mathbf{c}_t^{ij}$ ▷ Posterior (3.3.4)
 $\mathbf{p}_{t+1}^{ij} = \text{GRUCell}(\tilde{\mathbf{p}}_t^{ij}, \mathbf{d}_t^{ij})$
 $t \leftarrow t_{t+1}$
 end for
end for

B.4 Cost Learning

In this section, we talk about how we utilize the learned stable linearized dynamics of the system, to learn the quadratic cost matrices of LQR using expert demonstrations. We closely follow the cost learning method in (1), but instead of learning a different cost function and goal condition for each dynamic *mode*, modeled as a single discrete global variable, we learn a single cost function for a certain task with a fixed goal. Similar to (1), we aim to learn the *behavior* of the expert in a single scenario and generalize that behavior to new goal conditions.

We collect a set of 40 expert demonstrations $\{\mathcal{G}_0^*, \mathbf{u}_0^*, \dots, \mathcal{G}_T^*, \mathbf{u}_T^*\}$. Using the learned graph inference module, we first evaluate the edge activations $\{a_0^{ij}, \dots, a_T^{ij}\}$ over the expert trajectories. We check for the isolated nodes i.e. nodes that are not actuated and have no active edges connected to them in any of the trajectories at any time step and remove those nodes from all the expert graphs. This is based on the indication that these isolated nodes are not participating in the current task. Then, we append together the node attributes of participating nodes at each time step to get the full state of the system as a vector. These full state trajectories are then used as input expert demonstrations to the method described in (1) for learning the quadratic cost matrices Q and R . Since the input to LQR is the full appended state of the system, the diagonal terms in the learned Q and R matrices will correspond to penalties for being away from the goal and the non-diagonal terms will correspond to penalties for being away from other nodes. Thus, for tasks like picking up an object with an arm, the learned Q and R enable relative motion between the nodes and then motion towards the goal.

$$\begin{aligned}
 & \min_{\mathbf{u}_t} \sum_{t=0}^T \left((\mathbf{x}_t - \mathbf{x}_f)^\top \mathbf{Q}_t (\mathbf{x}_t - \mathbf{x}_f) + \mathbf{u}_t^\top \mathbf{R}_t \mathbf{u}_t \right) \\
 & \text{s.t. } \mathbf{x}_{t+1} = \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t, \quad i = 1, \dots, T \\
 & \quad \mathbf{x}_0 = \mathbf{x}_{\text{init}}
 \end{aligned} \tag{B.1}$$



APPENDIX FOR MREST

C.1 Environment Details

In this section we provide further details on the different environments used in our experiments.

C.1.1 MT-Coarse Manipulation

For coarse manipulation tasks we focus on a variety of objects including blocks, mugs, cups, and shoes (both men and women shoes). As noted in the main paper, for these set of objects we focus on pick-and-place skills. However, we note that we did experiment with more complex contact-rich skills (e.g. pushing, stacking). However, we found the physics to be unstable with more complex objects (e.g. cups). For instance, pushing cups would almost always topple them and roll over. For future work, we hope to make our skills more robust.

Specifically, we use fixed size blocks with different semantic colors, 4 mugs, 4 cups and 4 shoes. We use google scanned objects (156) to collect non-block objects and use mujoco (153) to simulate our environment. We use the latest mujoco environments to import meshes into the simulator. Each environment in this set of tasks is created by first selecting a target object-type and then selecting a target object from the set of objects. We then select 3-5 distractor objects to fill the scene. These objects are uniformly selected from the remaining objects.

C.1.2 MT-Precise Manipulation

As noted in the main paper for precise manipulation tasks we use the spatial precision set of tasks from RLBench (117). Overall, we use 4 tasks (see Figure 4.3 (Left)) – square block insertion, pick up small objects, shape sorting, and unplug usb from computer. We avoid using the motion-planner augmented approach for solving these tasks and instead opt for learning reactive closed-loop control policies. We use

the delta end-effector actions for our tasks. Additionally, we use standard front and wrist mounted camera. along with proprioceptive and force-torque feedback as policy input.

However, directly using end-effector actions increases the policy horizon significantly. Moreover, naively using the original input distribution for each task also requires learning full 6-DOF policies. Both of these can significantly increase the data requirements to learn the manipulation policy. To avoid this we restrict the starting distributions for each task such that the objects are spawned in a slightly narrow region in front of the robot. We further make other task-specific changes, detailed below, such that the robot can perform each task without changing hand orientations.

Insert Onto Square Peg: For this task we restrict the orientations of the square ring (blue object) and the peg on which to insert. This allows the robot to perform the task without changing gripper orientations. Further, we use a region of $40cm \times 30cm$ in front of the robot to spawn both the base and ring. Finally, the default task configuration provides 20 different peg colors, of which we use the first 10 colors for training and remaining 10 colors for robustness experiments.

Pick and Lift Small: For this task, we again use a region of $40cm \times 30cm$ in front of the robot to spawn both all objects. We also restrict the orientation of each object such that it can be grasped directly without requiring gripper orientation changes.

Shape-Sorting: The default configuration for the shape-sorting task considers 4 different shaped objects (see Figure 4.3 Bottom-Left) – square, cylinder, triangle, star, moon. In the default RLBench configuration most objects directly stick to the robot finger and are simply dropped into the hole for task completion. However, with closed loop control we find that non-symmetric objects (star, triangle, and moon) can have significant post-grasp displacement such that it is impossible to insert these objects without changing gripper orientation. Hence, we exclude these two objects from evaluation and only use symmetric square and cylinder objects.

Take USB Out: This task requires the robot to unplug a USB inserted into the computer. However, the default configuration for this task requires 6-dof control. To avoid this, we create smaller computer and USB assets and mount them vertically on the table such that the USB can be unplugged without changing hand orientation. See Figure 4.3 (Bottom-Right) for visualization.

C.1.3 MT-Dynamic Manipulation

This task involves using the CMU Ballbot in simulation (PyBullet (?)) to perform a dynamic pick up task. The task involves picking up a block that is placed on a table in front of the ballbot. We use two blocks (red and blue) in this task and use language instructions to specify which object to pick up. The initial conditions are set such that the table and objects are always out of the reach of the ballbot arms and the ballbot has to roll forward to pick up the objects. We use a statically

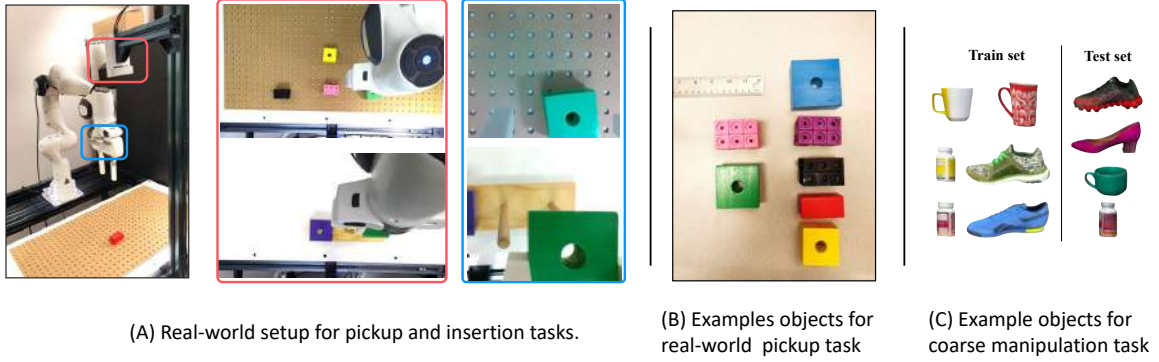


Figure C.1: *Left:* Real World env setup with third-person (red) and first-person (blue) camera views. *Middle:* Example objects set used for real-world pickup task. *Right:* Example objects used for MT-coarse.

mounted camera looking at the table and the ballbot as the third-person camera and the camera located on the turret of the ballbot as the first-person camera. The turret tilt is adjusted such that the objects on the table are initially out of the view of the turret camera and only when the ballbot starts moving towards the table, the objects come into view. The third person camera is always able to view both the objects and the ballbot. We use task space control to control the ballbot end-effector while a center of mass balancing controller is always running in a high-frequency feedback loop to balance the ballbot.

C.2 Architecture Details

Section 4.3 discusses the overall architecture used in our work. To recall, our proposed architecture uses a multi-resolution approach with multiple-sensors, each with different fidelity. We process each sensor with a separate network which is conditionally initialized using a pre-trained vision-language model. The output of each vision model is flattened to create a set of patches. For DETR (111; 112) based model we use a ResNet-101 backbone and flatten the output layer into 49 patches and add positional embedding to it. For CLIP (66) we use a ViT-B model and use hierarchical features from the 5'th, 8'th and 11'th layer. Since MDETR already does vision-language fusion using a transformer we directly use its output. However, since CLIP only weakly associates vision and language at the last layer, we additionally use FiLM layers to condition the output. Our use of FiLM is similar to previous models (?). For each camera modality we use a small transformer with multi-head attention. Each transformer uses an embedding size of 256 and 8 heads. We use post layer-norm in each transformer layer. Further, in each transformer layer we use cross-attention with the other camera. Overall we use 3 transformer layers for each camera modality.

Key	Value
batch size	16
proprio and force torque embedding	256
camera-transformer embedding Dim.	256
camera-transformer feedForward Dim.	768
Number of transformer layers	3
learning rate	0.0001
warmup epochs	5
total epochs	60
optimizer	AdamW
weight decay	0.01
scheduler	cosine

Table C.1: Hyperparameters used for our architecture and model training.

Our force-torque and proprioceptive input is concatenated together and mapped into 256 dimensions using a linear layer. We concatenate the readout tokens from each camera transformer and the force-torque embedding. This 256×3 size embedding is then processed by 2 linear layers of size 512 which output the robot action.

Input: For each of our camera sensor we use an image of size 224×224 . For proprioceptive input we use the end-effector position of the arm. While for force-torque input we use the 6 dimensional force-torque data. We use cropping augmentation for both camera sensors. Specifically, we first resize the image to 226 and then do random crop with shift = 8. For, more aggressive pixel level augmentations we stochastically apply grayscale and use color jitter with brightness $\in (0.4, 0.8)$, contrast $\in (0.4, 0.8)$, saturation $\in (0.4, 0.6)$ and hue $\in (0.0, 0.5)$. These augmentations significantly change the underlying visual semantics of the task.

C.2.1 Training Details

In this section we provide details on the demonstrations (for each environment type) used to train our approach. Further, we also provide details on the train and heldout configurations used for robustness evaluation.

MT-Coarse: As noted above in Appendix C.1.1, we use multiple different objects to train and evaluate our policy. Each environment is created by first sampling a target object and then a set of distractor objects. For each environment and skill combination we collect 20 demonstrations. Overall, this gives us ≈ 1000 demonstrations across all tasks. We then learn one policy across all tasks.

MT-Precise: For spatial precision tasks from tasks from RL Bench (117) we use 4 different tasks. As discussed in Section C.1.2, each task has it’s own set of variations. For training our multi-task policy we use try to balance the number of demonstrations from each task. For square peg insertion (*insert onto square peg*) task we use first 10 variations for training and gather 25 trajectories per variation. Each other task has less than 4 variations hence for each task we use 100 demonstrations each for

training. To test visual-semantic robustness for these tasks Section 4.5.2 we use the insert-onto-square-peg task since only this task has any semantic variations. We use the remaining 10 peg colors (i.e. 10 heldout variations) to test each approach.

MT-Dynamic: To collect expert demonstrations, we sample the locations of the objects on the table in a 70cm*20cm region and sample the initial ballbot location in a 50cm*50cm region. We collect 50 demonstrations for each task setting (each block). As noted earlier, the third-person camera is used at a frequency of 5Hz, the turret camera is used at 20Hz and proprioception and force-torque feedback is used at 75Hz.

Real-World: For real-world tasks we collect data using teleoperation with a leap-motion device which can track hand movements upto a 100Hz. We map these movements to robot movements and collect proprioceptive and force-torque data at 75Hz, while both cameras are recorded at 30Hz. To collect data for pickup tasks we use two blocks with different shapes and different colors. The green and pink blocks in Figure C.1 (Right) were used to collect all training data. While evaluation happened on 8 other blocks, each with a different shape and color. For training our policies we collect 60 demonstrations for each pickup variation and 50 demos for the insertion task. We note that the initial state distribution for insertion was narrower than pickup and hence it required fewer demonstrations.

Metrics: We use task success as the evaluation metric. Since we use a multi-task setting we report mean success over all tasks. During training, we evaluate the policy every 4 epochs on all train tasks. We report the average over *top-5* mean success rates across all evaluation epochs. For task generalization results (**Q3**) we use the trained policy and evaluate it on novel visual-semantic tasks which were never seen during training. Hence, for Q3 we report task success on novel unseen tasks. For all evaluations we use 20 rollouts per task. Further training details are provided in Appendix ??.

C.2.2 Implementation Details

In this section, we discuss our real-robot implementation details. In our implementation, the real-time control loop is composed of a low-level task-space impedance controller and a high-level neural network controller. The low-level controller operates at 1KHz using a real-time kernel and sends control commands to Franka Panda’s control interface (FCI) (35). Our neural-network controller implementation can operate up to a maximum of 100Hz given communication latency. Specifically, for our experiments we run the neural network controller at 75 Hz. We use fixed low impedance values (Kp: 350) to avoid damaging the robot during fast execution of contact-rich tasks.

Neural network controller implementation: For our real-robot neural-network controller implementation we follow a multi-threaded architecture. Robot state information such as proprioceptive data and force-torque data is published at 100Hz,

while camera images are published at 30Hz. Each sensor modality is appended to a separate fixed size time-stamped buffer. We process each modality independently in a multi-threaded manner by extracting the latest time-stamped data from the respective buffer.

Camera images are processed on separate threads using their respective neural networks and we save the network outputs for future processing. More specifically, we process images from third-person camera using a large VLM and save a set of visual-language representations from its output in a buffer. This thread is limited by the inference speed of the large VLMs and operates at 5Hz. We process the image from the in-hand camera in a separate thread using a small ResNet based model to get hand-camera image representations. On the same thread, we further process these hand-camera image representations with the existing cached vision-language representations using cross-attention layers to get multi-modal fused visual-language output which is added to a fixed size buffer. This thread operates at 20Hz.

Finally, the high-level neural network controller (which runs on the main thread at 75Hz) concatenates the cached robot state information (force-torque, proprioceptive) with the latest fused multi-modal features. The concatenated features are processed through a small multi-layer perceptron to get the final action output which is sent to the low-level impedance controller.

C.3 Additional Results

C.3.1 Additional Real-World Comparisons

In addition to real-world results in Table 4.5 we also tried out BC-Z and RT-1 on the pickup task in the real world. Table C.2 reports the average success rate and compares them to our method. We find that BC-Z’s performance is much worse than our proposed approach. This is because BC-Z operates at a single-resolution (both spatial and temporal) as it uses only a third-person camera. In the absence of a first-person camera view it is often unable to accurately localize the target object and fails to perform the final fine-grained motion to grasp the object and lift it up. Further, for RT-1 we find the performance to be very poor. We believe this is because RT-1 uses tokenized actions which requires us to discretize our continuous robot actions. Since we operate in the low data regime (120 trajectories) such discretization leads to token imbalances during training and deteriorates the model’s performance. Additionally, since RT-1, similar to BC-Z, uses single-resolution (i.e. third-person camera only) we believe its performance suffers from similar challenges of inaccurate localization. Furthermore, we evaluate visual generalization of the BC-Z and RT-1 policy on novel unseen objects (and instructions). Since both BC-Z and RT-1 do not use a pre-trained vision-language model and thus have no visual grounding for the text instructions they fail to perform well on unseen novel objects. By contrast, our approach that utilizes a pretrained VLM generalizes well.

Setup	BC-Z (64)	RT-1 (63)	Ours
Train	12.5	0.0	75.0
Eval	5.0	0.0	71.1

Table C.2: Real-World results for using commonly used imitation learning (single-spatial resolution baselines) for Pickup task.

	$\pi_{\text{low-res}}$	$\pi_{\text{high-res}}$	Ours
RealWorld - PegInsert	45.0	62.5	67.5

Table C.3: Additional Results for multi-temporal resolution experiments. As before, both $\pi_{\text{low-res}}$ and $\pi_{\text{high-res}}$ are single-resolution approaches which run at 5 Hz and 20 Hz respectively, while ours is a multi-resolution approach.

C.3.2 Additional Ablations

We further ablate on the different components of our proposed approach. For these set of results instead of using all 3 environment suites for evaluation, we choose the most appropriate environment suite for each component of our approach and evaluate on it.

Real-World Temporal-Resolution Comparison: We also ablate the effect of temporal resolutions on real-word robot performance. Specifically, we evaluate single temporal-resolution approaches ($\pi_{\text{low-res}}$) and $\pi_{\text{high-res}}$ for the peg-insertion task in the real-world. As before, to evaluate the learned policy we run each episode for a fixed duration of 60 seconds. However, we use early termination if the episode is solved successfully or the robot violates the desired workspace. Table C.3 shows our results. Given that the insertion task is not dynamic, $\pi_{\text{high-res}}$ performs similarly to our approach. However, by comparison, ($\pi_{\text{low-res}}$) performs much more poorly (45% only). This is because a low-temporal resolution policy is not very reactive and hence doesn’t respond fast to contacts made with the wooden peg. Thus, it is often unable to find the appropriate location to insert the block into the wooden peg. This can also be seen from qualitative videos (see success and failure videos), where both success and failure scenarios are much less reactive.

Temporal-Resolutions: Finally, we also ablate the temporal frequencies for the MT-Dynamic tasks. We ablate the effect of using camera inputs at low-resolution (third-person and in-hand camera inputs at 5Hz) while only force-torque feedback is used at high-resolution (75Hz).

Table C.4 below shows our results. From the table below, we observe that the performance on MT-Dynamic tasks drops significantly when using the camera views at a very low temporal resolution. From our qualitative observations we note two common failure cases. First, where the ballbot is sometimes unable to reach the block to pick up. This is because, due to latency in the camera inputs (5 Hz), the policy outputs sub-optimal actions. Upon receiving updated camera inputs the policy tries to correct the trajectory. The overall resulting trajectory is noisy and fails to reach the target object. Second, again due to camera latency, the end effector does

$\pi_{\text{low-res-high-FT}}$	Ours
33.4	73.6

Table C.4: Results for using low-temporal resolutions for camera-inputs (5Hz) and high-temporal resolutions for force-torque only (75Hz).

not align well with the target object and ends up toppling the object while trying to grasp it.

D

APPENDIX FOR VLTSafe

D.1 Background

D.1.1 Hamilton-Jacobi Reachability Analysis

Hamilton-Jacobi Reachability (HJR) analysis provides a formal approach for computing safe policies that guarantee constraint satisfaction by finding an optimal solution to a reach-avoid problem. Consider a discrete-time dynamical system $s_{t+1} = f(s_t, a_t)$, where t is the current time step, $s_t \in \mathcal{S} \subset \mathbb{R}^n$, $a \in \mathcal{U} \subset \mathbb{R}^m$, \mathcal{U} is compact and dynamics f are bounded and Lipschitz continuous. Target set $\mathcal{T} \subset \mathcal{S}$ describes *reach* states that satisfy $\{s : l(s) > 0\}$ and failure set $\mathcal{F} \subset \mathcal{S}$ describes *avoid* states that satisfy $\{s : g(s) < 0\}$. \mathcal{T} and \mathcal{F} are closed sets and $l(s), g(s) : \mathcal{S} \rightarrow \mathbb{R}$ are Lipschitz continuous functions. The safety problem is to find the *reach-avoid* set, $\mathcal{RA}(\mathcal{T}; \mathcal{F})$, which is the set of states from which a controller can drive the system to \mathcal{T} while avoiding \mathcal{F} at all times t . It was demonstrated in (142) that states belonging to the set $\mathcal{RA}(\mathcal{T}; \mathcal{F})$ satisfy $V(s) > 0$, where $V(s)$ is the solution to the following fixed-point Reach-Avoid Bellman equation (RABE):

$$V(s) = \min \left[g(s), \max \left(l(s), \sup_{a \in \mathcal{U}} V(f(s, a)) \right) \right] \quad (\text{D.1})$$

$$V(s) > 0 \Leftrightarrow s \in \mathcal{RA}(\mathcal{T}; \mathcal{F})$$

It is important to note that, unlike the Bellman updates in dynamic programming based approaches (191), (D.1) does not have a time-discounting term, thus it does not induce a contraction mapping and cannot converge to a fixed point using value iteration.

D.1.2 Reach-Avoid Reinforcement Learning (RARL)

Based on time-discounting in temporal difference learning (144) and Q-learning (145) based methods, (142) introduced a time-discounting term in RABE (D.1), thus

inducing contraction mapping in value function learning, which was extended to the reach-avoid setting and deep Q-learning by (124). The Discounted Reach-Avoid Bellman Equation (DRABE) (124) can be written as:

$$V(s) = \gamma \min \left[g(s), \max \left(l(s), \sup_{a \in \mathcal{U}} V(f(s, a)) \right) \right] + (1 - \gamma) \min \left(l(s), g(s) \right)$$

where γ is the discount factor, which can be interpreted as the probability of episode continuation.

D.1.3 Offline Parameterized Safe Policy Learning

We utilize Deep Deterministic Policy Gradient (DDPG) (155) as our reinforcement learning framework to learn a parameterized safe policy in continuous action spaces. The Q-function update is modified to the discounted reach-avoid formulation (D.2). Suppose Q_ϕ is the Q-network, μ_θ the policy network, and $Q_{\phi_{\text{targ}}}$ and $\mu_{\theta_{\text{targ}}}$ are the target Q-network and target policy networks respectively. Given a sample (s, a, r, s', d) from the replay buffer \mathcal{D} , the Mean-squared Bellman Equation (MSBE) in DDPG is as follows:

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left[\left(Q_\phi(s, a) - \left(r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \right) \right)^2 \right]$$

where $r = \min(l(s), g(s))$ is the reward and d indicates whether state s' is terminal. The Discounted Reach-Avoid Bellman Error for the DDPG algorithm can be written as:

$$\begin{aligned} L(\phi, \mathcal{D})_{\text{DRABE}} = & \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left[Q_\phi(s, a) - \left[(1 - \gamma) \min(l(s), g(s)) \right. \right. \\ & \left. \left. + \gamma \min \left\{ g(s), \max \{ l(s), Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \} \right\} \right] \right] \end{aligned}$$

D.2 Experimental details

We denote the *full state* of the robot and the objects via $s = [s^{\text{EE}}, s^{\text{cbot}}, s^{\text{ctop}}, s_1^o, \dots, s_N^o] \in \mathcal{S}$ where \mathcal{S} is the full state space, s^{cbot} is the state of the bottom cereal box, s^{ctop} is the state of the top cereal box and s_i^o is the state of the i^{th} object in clutter on the table. We model the robot’s state as the position and velocity of the end-effector $s^{\text{EE}} = [x^{\text{EE}}, \dot{x}^{\text{EE}}]$ where $x^{\text{EE}} \in \mathbb{R}^3$ is the Cartesian position and $\dot{x}^{\text{EE}} \in \mathbb{R}^3$ is the velocity. Let each object’s state be represented by $s_i^o = [x_i^o, \dot{x}_i^o], i \in \{1, \dots, N\}$ where N is the total number of cluttered objects on the table, that is, in addition to the two cereal boxes.

	EE	cbot	ctop	O_1	...	O_N
EE	I	I	I	m_1^{EE}	...	m_N^{EE}
cbot	I	I	I	m_1^{cbot}	...	m_N^{cbot}
ctop	I	I	I	m_1^{ctop}	...	m_N^{ctop}
O_1	m_1^{EE}	m_1^{cbot}	m_1^{ctop}	m_1^1	...	m_N^1
\vdots	\vdots	\vdots	\vdots	\vdots		\vdots
O_N	m_N^{EE}	m_N^{cbot}	m_N^{ctop}	m_1^N	...	m_N^N

Figure D.1: Custom attention mask

Hyperparameter	Value
Batch size	200
# Training steps	3M
Q learning rate	1e-4
Policy learning rate	1e-4
Max. episode length	300
Optimizer	AdamW
Eps	1e-8
Weight decay	0.01
Clip grad norm	5
Replay buffer size	100k
Discount factor	0.99
Polyak	0.995
Action noise	0.1
Collision threshold	1 cm

Table D.1: Training hyperparameters

D.2.1 Safety and Target Representation

The safety constraints capture how the stack of boxes should not topple too far when the robot is pulling the cereal box from underneath, and models robot-object interactions. Specifically, \mathcal{F} reasons about the following constraints: 1) at any time step t , the top cereal block should not be excessively displaced from its initial position $g_{\text{stack}}(s_t) = d_{\text{thresh}} - \|x_{t=t}^{\text{ctop}} - x_{t=0}^{\text{ctop}}\|$. For instance, if the bottom box is pulled very aggressively, the top cereal box will be flipped over or if the bottom block is pulled too slowly, the top block will move with it due to friction. Both these interactions should be avoided; 2) the bottom cereal box should not make contact with any of the *fragile* objects $g_{\text{fragile}}(s) = \|x^{\text{cbot}} - x_{\text{fragile}}^{\text{o}}\| - d_{\text{thresh}}$; 3) the bottom cereal box should make soft contact with the *soft* objects $g_{\text{soft}}(s) = d_{\text{thresh}} - \|\dot{x}^{\text{cbot}} - \dot{x}_{\text{soft}}^{\text{o}}\|$. This is a relative velocity constraint, which requires long-horizon reasoning, so that the cereal box can slow down before making contact with the object; 4) the bottom cereal box should not move over *sensitive* objects $g_{\text{sensitive}}(s) = \|x^{\text{cbot}}[:2] - x_{\text{sensitive}}^{\text{o}}[:2]\| - d_{\text{thresh}}$ i.e. their x-y positions should not overlap; 5) any contact is allowed with *durable* objects $g_{\text{durable}}(s) = +\text{inf}$. Note that since the safety constraint depends on the semantic properties of the object in the scene, during deployment the VLM specifies the constraints types via the parameterization $\phi \in \{\text{no-contact}, \text{soft-contact}, \text{any-contact}, \text{do not move over}\}$. During training, we consider all combinations of object properties and respective constraints.

The target set represents the desired cereal box reaching the correct goal location. Specifically, \mathcal{T} always requires 1) the bottom cereal box to be completely separated from the top cereal box $l_{\text{slide}}(s) = \|x^{\text{cbot}} - x^{\text{ctop}}\| - d_{\text{thresh}}$, and 2) placed in one of two regions near the far end of the table $l_{\text{goal}}(s) = d_{\text{thresh}} - \|x^{\text{cbot}} - x^{\text{goal}}\|^{1,2}$. This induces two possible target sets that the VLM can choose from at deployment time via the

target parameterization $\tau \in \{\text{top-goal}, \text{bottom-goal}\}$. During training, we randomly choose between the two goal regions.

D.2.2 Custom attention mask

The custom attention mask used in **VLTSafe** is illustrated in Fig. D.2.2. Each element m_j^i of the mask indicates whether object i attends to object j ; that is, $m_j^i = 1$ if object i attends to object j , and $m_j^i = 0$ otherwise. As noted in Fig. D.2.2, the end-effector (s^{EE}), bottom cereal box (s^{cbot}) and top cereal box (s^{ctop}) attend to each other. Specifically, $m_j^{\text{EE}} = 1$ if object j is among the top- k nearest objects to the end-effector. The same logic applies for the other pairs.

D.2.3 Architecture and training details

All policies are trained using the same target and failure sets as defined in Sec. D.2.1 and optimized via the update rule discussed in Sec. D.1.3. The training hyperparameters are kept consistent across all methods and are shown in Table D.1. The policies are trained using one NVIDIA RTX A6000 GPU and an Intel i9-10900K CPU.

In **VLTSafe**, both the actor and critic networks are composed of their respective tokenizers, multi-head self-attention layers and output heads. Each tokenizer comprises two hidden layers with 64 units each and SiLU activation, followed by an output layer that projects inputs into a 128-dimensional token embedding space. The core transformer backbone follows the standard architecture of (author?) (154), with two layers of multi-head self-attention, each employing four attention heads, as shown in Fig. D.2. The feedforward block is composed of a single hidden layer of size 256 and an output layer that projects back to the token embedding dimension. The action and value prediction heads are implemented as two-layer MLPs with 64 hidden units per layer. In total, **VLTSafe** contains approximately 620k trainable parameters. The actor and critic networks in the **MLP** baseline are implemented simply as five-layer MLPs with 256 hidden units per layer and ReLU activation resulting in 540k total trainable parameters.

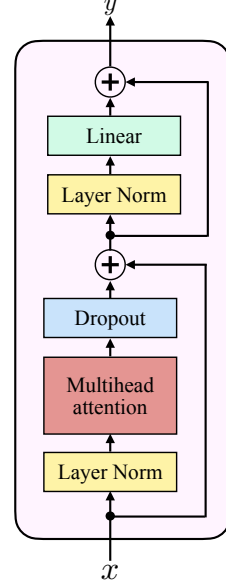


Figure D.2: **Attention block**

D.3 VLM for Test-Time Safety and Target Specification

Agent role prompt: You are an excellent safe planning agent for dynamic tasks. You are given a task description and an image showing the robot and objects on a table. The robot is trying to slide the blue box from under the red box and to the goal regions to the right of the table, without damaging other objects along the way.

Safety constraint selection prompt: Each object on the table can potentially come in contact with the end-effector. You need to decide the safe interaction type for each object on the table from the list of constraint types. Here the description of the constraint types: **no_contact** implies that there should absolutely be no contact with a certain object. **soft_contact** implies that you can softly interact with that object, push it softly, etc. **any_contact** implies that any kind of interaction including aggressive impact is allowed. **no_over** implies that the robot is not allowed to move over (on top of) the object. Some hints on how to decide on the constraint type for an object: If an object is soft or made of durable material, and softly pushing it or moving it without toppling it is okay, **soft_contact** can be allowed with that object. If an object is very durable, and pushing it aggressively will not damage it, **any_contact** can be allowed with that object. If an object is fragile, and contacting it might damage it, **no_contact** should be allowed with that object. If an object is very sensitive like an open laptop or a bowl of food, and moving over it might be risky, **no_over** should be constrained for that object. Usually objects such as cups, wine glasses, bowls, electronics, etc are considered fragile and should be **no_contact**. Plastic objects such as bottles, plastic cans, tubes can be allowed **soft_contact**. Soft and non-critical objects such as toys, clothing, etc. are soft and can be ignored and allowed **any_contact**. Provide brief explanation, for choosing a specific constraint type for an object. In **image_description** briefly describe the scene and features relevant to the task.

Target constraint selection prompt: The robot is trying to slide the blue box from under the red box and place it in one of the two goal regions to the right of the table indicated by the two yellow squares on the table. The square closer to the robot is **bottom_goal** and the square further away is **top_goal** . Choose the target region for the blue box from the two goal regions such that it is safe to slide the blue box to that target region. Also provide a brief explanation for choosing that target region.

Structured Output We employ the structured output capabilities of OpenAI’s Python API to force a desired structure on what is output by GPT4o. Below is the `create_vlm_constraint_response` function used in the implementation of **VLTSafe**

```
1 def create_vlm_constraint_response(object_list,
2   safety_constraint_list, target_constraint_list):
3
4     fields = {}
5     for i in range(len(object_list)):
6         fields[f"explanation_obj_{object_list[i]}"] = (str, ...)
7         fields[f"{object_list[i]}"] = (safety_constraint_list, ...)
8
9     SafetyConstraintsPerObject = create_model("
10        SafetyConstraintsPerObject", **fields)
11
12     class ConstraintResponse(BaseModel):
13         safety_constraints_per_object: SafetyConstraintsPerObject
14         target_region: target_constraint_list
15         target_description: str
16         image_description: str
17
18     return ConstraintResponse
```

Code Listing D.1: The `create_vlm_constraint_response` function used to structure output from GPT4o.

The `create_vlm_constraint_response` function takes as input the list of objects in the scene, list of semantic targets (`bottom_goal`, `top_goal`) and the list of valid safety constraints for each object (`no_contact`, `soft_contact`, `any_contact`, `no_over`). These lists are used to populate the member variables of `ConstraintResponse` ensuring that both target and safety constraint outputs are restricted to valid options when querying the OpenAI API, thereby helping to prevent hallucinations.

D.4 Real-world Experiments

We control the Franka robot using Cartesian impedance control, which takes the desired end-effector pose as input at each time step. Our learned policy outputs the desired displacement of the end-effector, which is converted into an absolute pose and passed to the controller. Real-world rollouts for increasing number of objects than seen during training are shown in Fig. D.3

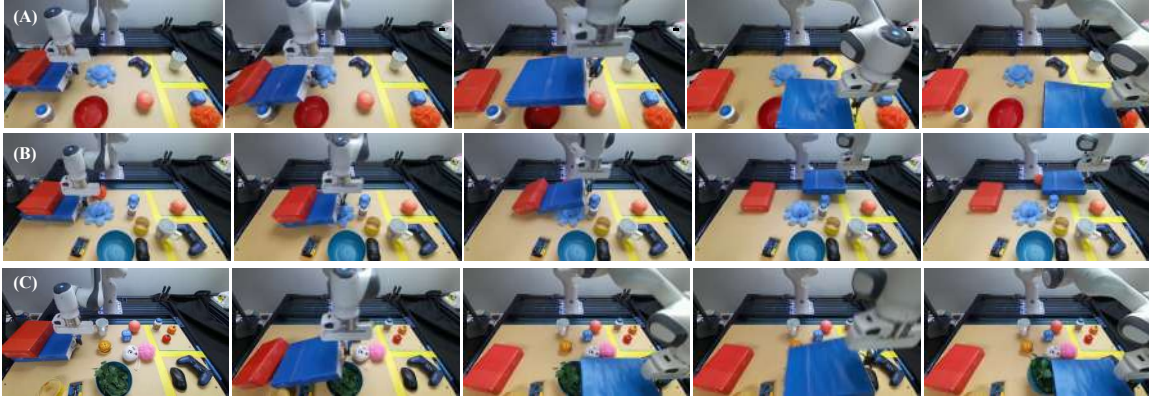


Figure D.3: **Real-world deployment:** Zero-shot sim-to-real transfer of a policy trained in simulation (*RandomConst*, 6 objects) to a real-world settings with (A) 8, (B) 10 and (C) 14 objects. Constraints are first inferred using a VLM. In (A) and (B) the VLM is asked to specify constraints based on common sense reasoning while in (C) the VLM is directed to strictly avoid food shaped objects (see Sec. D.4.1). As the robot pulls the blue box from under the red one, it makes allowable contact with the soft/durable objects, moves around or lifts up to avoid fragile objects and places the box safely in the goal regions deemed safe by the VLM.

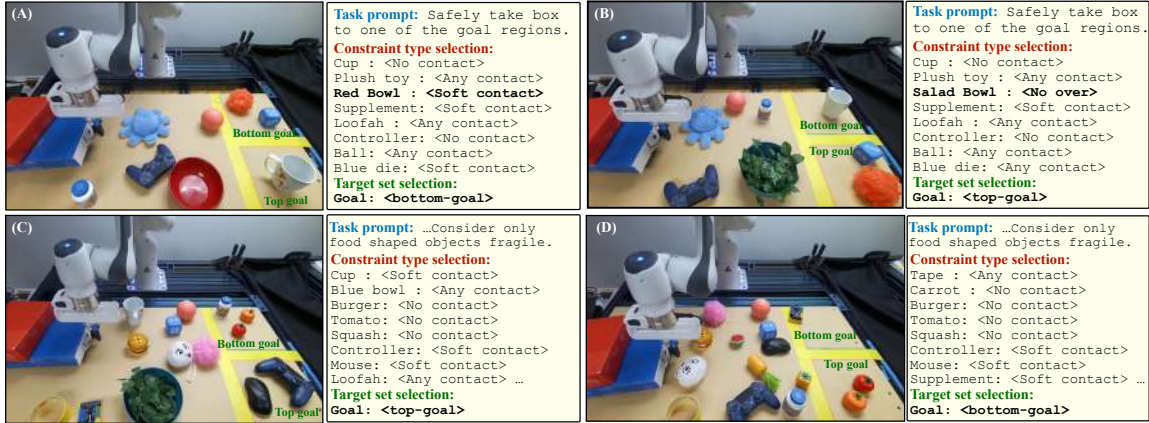


Figure D.4: Qualitative results for the VLM selecting safety constraint types and target regions based on the semantics of the objects as observed in the input image as well as the task description.

D.4.1 Qualitative examples of VLM constraint selection

In Fig. D.4(A) and (B), the VLM is prompted to specify target regions and safety constraints based on commonsense reasoning about the material properties of objects. As a result, electronics and porcelain cups are treated as fragile. Accordingly, the goal region in both (A) and (B) is selected to avoid including the porcelain mug. When the bowl contains salad, as in (B), it is assigned the constraint **no_over**; otherwise, as in (A), it is assigned **soft_contact**.

In Fig. D.4(C) and (D), the VLM is given explicit instructions to treat certain objects in specific ways. In this case, it is directed to consider only food-shaped objects

as fragile. Consequently, these objects are assigned the constraint **no_contact**, and the goal regions are selected to avoid them. Notably, electronics previously considered **no_contact** in (A) and (B) are now assigned **soft_contact** in (C) and (D), in accordance with the user’s updated instructions. This highlights the VLM’s capacity to interpret user intent and adapt task constraints accordingly, which is particularly valuable for nuanced and complex tasks requiring human-in-the-loop guidance.

E

APPENDIX FOR GRAPHEQA

E.1 Habitat environment setup

The Habitat-Sim setup for our experiments is identical to the setup used in (165). The camera sensor settings are as follows: image width = 640, image height = 480, camera height = 1.5m, camera tilt = -30 degrees, field of view = 120 degrees. For generating trajectories for the `<Goto_Object_node>(object_id)` and `<Goto_Frontier_node>(frontier_id)` actions, we find the shortest path between the current agent position and the desired object/frontier node location, on the obstacle-free voxel space of the 2D occupancy map. We orient the agent such that camera always points towards the desired node location all along the trajectory. In case of the `<Goto_Object_node>(object_id)` action, this maximizes the number of views that capture the target object. In case of the `<Goto_Frontier_node>(frontier_id)` action, this makes the agent look outwards into the unexplored areas.

E.2 HM-EQA and OpenEQA Datasets

The Habitat-Matterport Embodied Question Answering (HM-EQA) dataset introduced by Yadev et al. (176) is based in the Habitat-Matterport 3D Research Dataset (HM3D) of photo-realistic, diverse indoor 3D scans (192). The dataset is composed of 500 multiple choice questions from 267 different scenes which fall in the following categories: identification, counting, existence, state, and location. The OpenEQA dataset (175) contains data spanning the categories spatial understanding, object state recognition, functional reasoning, attribute recognition, world knowledge, object localization, and object recognition. We use both HM-EQA and OpenEQA to benchmark our results against baselines in simulation.

E.3 Hydra 3D Scene Graphs

Hydra 3DSGs are comprised of the following layers: *Layer 1 (bottom)*: a metric-semantic 3D mesh, *Layer 2*: objects and the agent, *Layer 3*: regions or places, *Layer 4*: rooms, and *Layer 5 (top)*: building. Intra-layer edges between nodes denote ‘traversability’, while inter-layer edges denote ‘belonging’. For example, an edge between regions in Layer 3 denotes traversability between these regions and an edge between an object and a room denotes that the object is located in that room. 3DSGs are constructed using RGB and depth images from the robot’s camera, camera pose and camera intrinsics. Using off-the-shelf image segmentation models (182), the object nodes are assigned semantic object labels.

E.4 Prompting

E.4.1 GPT Prompt

The full prompt provided to GPT4o for GraphEQA is given in Appendix E.12.2. In it we provide the scene graph description, description of the agent’s current state, agent prompt, and just generally more descriptive text for more context.

E.4.2 Hierarchical Nature of 3DSGs and Planning

The portion of the prompt used to describe the scene graph in GraphEQA clarifies to the VLM how layers and nodes are organized in a 3DSG. We take advantage of this structure by requiring `<Goto_object_node_step>` to be organized such that the VLM first chooses a room (level 4) to navigate to, then choosing an object (level 2) in that room. This inherent structure and explanation of it in the prompt guides the VLM to choose actions that investigate objects that are semantically relevant to the question.

E.5 Baseline Performance on Task Categories

Table E.1 shows the performance of baselines and GraphEQA across the different task categories in the HM-EQA dataset. GraphEQA outperforms all other methods across all task categories, but is particularly more performant in comparison when considering Counting and State tasks. It is worth noting that the Counting and State categories are among the most challenging.

Table E.1: Success Rate (%) in simulation for task categories in the HM-EQA dataset, for Identification, Counting, Existence, State, and Location categories. Reported in terms of category successes / total number of category EQA tasks. [†] indicates our implementation of that baseline.

Method	Ident.	Counting	Existence	State	Location
Explore-EQA	59.2	46.2	56.5	46.5	47.7
Explore-EQA-GPT4o [†]	32.5	44.2	56.4	42.3	40.8
SayPlanVision [†]	75	44.4	63.3	43.4	56
GraphEQA	77.8	57.9	69	65.2	64

E.6 Additional Ablations:

We perform some additional ablations to evaluate the utility of different components of our method: **GraphEQA-NoEnrich**, which does not use frontier enrichment (Section 6.3.2), and **GraphEQA-CurrView**, which uses only the current view as input to the VLM and does not choose additional task-relevant keyframes (Section 6.3.3). All ablations of GraphEQA use GPT-4o. Here we analyze these two additional ablations, **GraphEQA-NoEnrich** and **GraphEQA-CurrView**. We observe that GraphEQA-NoEnrich performs slightly worse than GraphEQA which demonstrates that enriching the scene graph with additional semantic information in the form of edges between frontiers and nearest objects, as discussed in Section 6.3.2, lends itself to semantically informed exploration. We observe that the performance drop is worse in the case of GraphEQA-CurrView, where we do not use task-relevant visual memory, but only the current view of the agent. This demonstrates that task-relevant visual memory is very useful in long-horizon tasks where the current view of the robot might not be the best view for answering an embodied question.

Table E.2: Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.

Ablation	Succ. Rate (%)	#Planning steps	L_τ (m)
GraphEQA-NoEnrich	59.5	5.1	11.1
GraphEQA-CurrView	53.1	5.7	12.2
GraphEQA	63.5	5.1	12.6

E.7 Additional Quantitative results: Zero-shot reasoning for commonsense questions

We perform this evaluation to answer the following question: how many questions in the HM-EQA dataset can be answered *correctly* purely based on commonsense reasoning or guessing, without exploring the environment? We aim to perform this

Table E.3: Additional baselines (Simulation): Success rate (%)

Method	Succ. Rate (%)
Explore-EQA (165)	51.7
Explore-EQA-GPT4o	46.4
SayPlanVision	54.8
GraphEQA	63.5
EQAZeroShotGPT4o	17.2
EQAZeroShotPrismatic	1.8
EQAZeroShotGPT4oQuestionOnly	6.6

analysis to roughly estimate the percentage of false positives that can occur in the HM-EQA dataset using different VLMs. To answer this question we define the following three additional baselines: **EQAZeroShotGPT4o**, **EQAZeroShotPrismatic** and **EQAZeroShotGPT4oQuestionOnly**. **EQAZeroShotGPT4o** evaluates the zero-shot performance of GPT-4o when answering an EQA question using the initial image and the question. **EQAZeroShotPrismatic** evaluates the zero-shot performance of the calibrated Prismatic model from Explore-EQA (165) when answering an EQA question using the initial image and the question. **EQAZeroShotQuestionOnly** evaluates the zero-shot performance of GPT-4o when answering an EQA question using only the question. In all the above baselines, no exploration steps are taken. Prompts for the above baselines are identical to ones used by Explore-EQA (165). An episode is considered a success if the question is answered correctly and with high confidence (> 0.5).

Table E.3 shows the simulation results for the baselines mentioned above compared to the baselines discussed in Section 6.4.1. We observe that, given only the question, **EQAZeroShotQuestionOnly** answers 6.6% of the questions correctly with high confidence. This can be attributed to the VLM answering questions based on commonsense reasoning or even just random guessing, and getting them correct. **EQAZeroShotPrismatic** answers only 1.8% of the questions correctly with a confidence > 0.5 . This can be attributed to the fact that the Prismatic VLM is calibrated to avoid overconfident behavior in the absence of evidence, hence the zero-shot confidence values are low. **EQAZeroShotGPT4o** answers 17.2% of the questions correctly with high confidence. To evaluate whether these questions were answered based on actual evidence in the initial image or purely based on commonsense reasoning/guessing, we further qualitatively evaluated the successful cases. Among the 17.2% that **EQAZeroShotGPT4o** answers successfully, 8.8% were answered based on actual evidence in the initial image. These questions could be answered using the initial image. The remaining 8.4% were answered based on commonsense reasoning/guessing without any evidence from the environment. Thus, 8.4% is the rough estimate of the false positives that can occur in the HM-EQA dataset using GPT-4o

based methods.

E.8 Error Analysis of Competing Baselines

Given the nature of the EQA tasks, it is possible that some of the questions are answered successfully using only commonsense reasoning/guessing, without grounding the answer in the current environment. We consider these cases as **false positives**. An example of a false positive is shown in Figure E.1. Furthermore, we also notice **false negatives**, where the answer was marked incorrect given the answer in the data set, although given the current image and scene graph, the answer seemed appropriate. Such cases exist due to ambiguities in the dataset. An example of a false negative is shown in Figure E.2. To get an estimate of the number of false positives and false negatives in our baselines, we uniformly sample a set of 114 questions from the HM-EQA dataset and manually label the results across the four categories: True Positives, True Negatives, False Positives and False Negatives. The results are shown in Table E.4 where each number is a percentage of the total number of questions considered (114).

Table E.4: Error analysis (Simulation): Percentage %

	GraphEQA	Explore-EQA	Explore-EQA-GPT4o
True positive	58.18	31.58	22.81
True negative	31.82	44.74	46.49
False positive	6.36	16.67	24.56
False negative	3.64	7.02	6.14

From Table E.4, we observe that GraphEQA has the least number of false positives and false negatives, yielding more reliable success rates. We note that **Explore-EQA-GPT4o** has a considerable percentage of false positives, i.e., questions are answered correctly based on guessing without grounding the answer in the current environment. This sheds light on why **Explore-EQA-GPT4o** has comparable success rates to **Explore-EQA-GPT4o**, even with considerably fewer planning steps (Table 6.1). This provides further evidence that GraphEQA effectively grounds GPT-4o in the current environment, is not overconfident based on commonsense reasoning and explores the environment until it can answer the question based on evidence. See additional results in Appendix E.5, categorized by question type.

App. E – Appendix for GraphEQA

Question: Did I leave any pot on the stove?

A. Two B. None C. Three D. One

Answer: B



True Positive



False Positive

Figure E.1: An example of a false positive case. The image on the left is the image that can be used to answer the question correctly. The image on the right is the image used by an agent to 'guess' the answer correctly with high confidence without grounding the answer in the current environment.

Question: Which pillows are there on the bed right now?

A. Green ones B. Black ones C. Red ones D. Purple ones

Answer: D



True Positive



False Negative

Figure E.2: An example of a false negative case. The question inquires about the color of the pillow on the bed. The question is ambiguous. On the left is the image that corresponds to the answer in the dataset i.e. purple pillows. On the right is an image that the agent encounters during exploration and answers that the pillows are 'green' with high confidence. Given the image, the answer is correct but is deemed incorrect in the dataset.

E.9 Structured Output

We employ the structured output capabilities of OpenAI’s Python API to force a desired structure on what is output by GPT4o. Below is the `create_planner_response` function used in the implementation of GraphEQA.

```

1  def create_planner_response(frontier_node_list, room_node_list,
2      region_node_list, object_node_list, Answer_options):
3
4      class Goto_frontier_node_step(BaseModel):
5          explanation_frontier: str
6          frontier_id: frontier_node_list
7
8      class Goto_object_node_step(BaseModel):
9          explanation_room: str
10         explanation_obj: str
11         room_id: room_node_list
12         object_id: object_node_list
13
14     class Answer(BaseModel):
15         explanation_ans: str
16         answer: Answer_options
17         explanation_conf: str
18         confidence_level: float
19         is_confident: bool
20
21     class PlannerResponse(BaseModel):
22         steps: List[Union[Goto_object_node_step,
23             Goto_frontier_node_step]]
24         answer: Answer
25         image_description: str
26         scene_graph_description: str
27
28     return PlannerResponse

```

Code Listing E.1: The `create_planner_response` function used to structure output from GPT4o.

The `create_planner_response` function takes as input enums for frontier nodes, room nodes, region nodes, object nodes, and the answer options for the particular question being answered by the VLM. These enums are used to populate the member variables of the `Goto_frontier_node_step`, `Goto_object_node_step`, and `Answer` classes, enforcing both type as well as the options available when calling the OpenAI API.

E.10 Additional Real-world Experiments in Home Environments

Additional experiments for Home Environment (a) and the questions asked of GraphEQA are shown below. We provide a sequence of ten images from the head camera on the robot to illustrate exploration and validation of GraphEQA’s answer to the question. Each experiment for Home Environment (a) was executed twice successfully. We show a single trial from this set for each experiment.

E.10.1 Home Environment (a)

Question 1: *Is there a blue pan on the stove?*

A. Yes

B. No

Answer: **A**



Figure E.3: **Home Environment (a) Question 1: Is there a blue pan on the stove?** For this question, the agent takes four VLM steps in the environment, beginning by reasoning about the current information it has access to; “I can’t answer confidently as the current view and scene graph don’t reveal a kitchen or stove.”. The agent then takes two GotoFrontierNodeStep steps to explore, and as shown in Frame 5 (from the left) gains access to the stove via the scene graph: “The stove is connected to region and frontier nodes, suggesting proximity.”. The agent then takes a GotoObjectNodeStep(stove) step, navigates to the stove, and upon reaching it answers “**The image shows a blue pan on the stove.**”, answering with high confidence. To see the full experiment in video, please see grapheqa.github.io.

Question 2: *How many white cushions are there on the grey couch?*

A. 1

B. 2

C. 3

D. 4

Answer: **B**



Figure E.4: **Home Environment (a) Question 2: How many white cushions are there on the grey couch?** For this question, the agent takes a total of six VLM steps, initiating its exploration by reasoning about the frontiers and objects it has access to in the scene graph; “No frontiers or objects are directly related to locating the grey couch. I’ll choose a frontier to explore unexplored areas.”. The agent then has access to a couch (see Frame 4 from the left) and chooses GotoObjectNodeStep(couch) for two VLM steps to search around the only couch it can see. After executing these object node steps, the agent then has access to a secondary couch it has not explored, and so chooses GotoObjectNodeStep(couch) once more to explore the grey couch, stating “The current view doesn’t clearly show the number of white cushions on the grey couch, so I need a closer look.” In the final VLM step the agent answers the question; **“The image shows a grey couch with two white cushions. There is also a table with various items in front of the couch.”** To see the full experiment in video, please see grapheqa.github.io.

Question 3: *Where is my handbag?*

- A. On the coat rack
- B. On the floor
- C. On the couch

Answer: **A**



Figure E.5: **Home Environment (a) Question 3: Where is my handbag?** The agent takes a total of three VLM steps to determine the location of the handbag, beginning with exploration guided by the scene graph; “The scene graph shows several frontiers connected to objects or areas, but none seem directly related to a potential couch or coat rack. Exploration of frontiers is necessary”. After the first GotoFrontierNodeStep to explore, the agent identifies a handbag in the environment; “Although the scene graph indicates the handbag is in the living room, further exploration is needed to confirm its position.”, but cannot yet confirm its position among the options available to it. Finally, the agent takes a GotoObjectNodeStep to gain a better view of the handbag and its position; **“I have visual confirmation from the current image showing the handbag on the coat rack.”**. To see the full experiment in video, please see grapheqa.github.io.

Question 4: *Where is the trashcan?*

- A. Next to the sink
- B. Next to the refrigerator

Answer: **A**



Figure E.6: **Home Environment (a) Question 4: Where is the trashcan?** The agent takes a total of six VLM steps to determine the location of the trashcan, beginning with exploration guided by the scene graph; “The image and scene graph don’t provide information about a kitchen or a trashcan location. Choosing a frontier that might lead to a kitchen.” The following four steps are GotoFrontierNodeStep actions, as the agent prioritizes exploring over investigating specific objects in the environment; “The current scene graph shows objects like a cushion, stove, chair, table, blanket, and other kitchen-related items like a dishwasher, hand towel, cabinet, and sink in the vicinity.” Finally, the agent visually confirms the location of the trashcan; **“Given the clear visibility of the trashcan next to the sink in Image 1, I am confident in answering with certainty.”** Note that *Image 1* for this experiment is the 9th image in the sequence of ten above. To see the full experiment in video, please see grapheqa.github.io.

Question 5: *Is the front door next to the staircase open?*

A. Yes

B. No

Answer: **B**



Figure E.7: **Home Environment (a) Question 5: Is the front door next to the staircase open?** The agent takes a total of four VLM steps to determine the state of the door, beginning its exploration by identifying where it is located and choosing an action appropriately; “The scene graph shows the agent’s location in the kitchen. There aren’t any nodes indicating a door or staircase. I will explore a frontier as it could lead to a new area where the front door or staircase might be visible”. The agent’s next two actions are GotoFrontierNodeStep actions, explaining the second of these choices as follows; “To potentially locate the staircase, I will explore the frontier connected to a couch, which could be in the living room area.”. The agent then chooses another frontier since “...the current view does not show much that relates directly to the front door or staircase.” Finally, along its trajectory to that frontier node, the agent identifies relevant images to the question and answers; **“The current image shows a blue couch in the center. To the left, a staircase is visible. Just beyond the couch is a door that is closed.”**. To see the full experiment in video, please see grapheqa.github.io.

E.10.2 Home Environment (b)

Question 1: *What is the color of the dehumidifier?*

A. Blue

B. White and Gray

Answer: **B**



Figure E.8: **Home Environment (b) Question 1: What is the color of the dehumidifier?**

The agent takes a total of nine VLM steps to answer this question, and starts by exploring a frontier node; “I need to find the dehumidifier machine to determine its color. It is not currently in the scene graph or visible.” The next eight steps are chosen as frontier steps, with explanations like “The dehumidifier is not in the scene graph. I need to explore to find it. The current image shows a potential candidate object (white appliance) near the chair (object3) and backpack (object17).” The final frontier step taken reveals the dehumidifier; “The object potentially representing the dehumidifier (object26, labeled ‘box’) is in room0. There are two white, boxy machines on the floor, likely dehumidifiers or air purifiers. One has a gray top panel.” To see the full experiment in video, please see grapheqa.github.io.

Question 2: *Is there any object on the black plastic drawer?*

A. Yes

B. No

Answer: **B**



Figure E.9: **Home Environment (b) Question 2: Is there any object on the black plastic drawer?**

The agent takes a total of nine VLM steps. The first step taken is a frontier node step chosen due to its connection to a candidate object; “I need to explore the environment to find the ‘black plastic drawer’ mentioned in the question, as it is not currently in the scene graph. The image shows a potential candidate object near the ‘chair’ (object1). Frontier3 is linked to the ‘chair’ (object1) and appears to be in the direction of the area needing investigation based on the current view.” The following six steps are GotoFrontierNodeStep actions with explanations similar to “This object is not currently in the scene graph or visible. Exploring a frontier might reveal new areas and objects, potentially including the drawer.” Finally, the agent is able to visually confirm the presence of the black plastic drawer with high confidence. To see the full experiment in video, please see grapheqa.github.io.

Question 3: *What is next to the white shopping bag?*

A. Stool

B. Broom

C. Dehumidifier

Answer: **A**



Figure E.10: **Home Environment (b) Question 3: What is next to the white shopping bag?** The agent takes a total of four VLM steps in this experiment, all of which are exploratory GotoFrontierNodeStep actions until the stool is identified as being next to the shopping bag.

E.11 Additional Real-world Experiments in Office and Workshop Environments

Additional experiments in the office environment are shown in Figure E.12 and Figure E.13. They include both success and failure cases. For videos of additional experiments in the home environment, please refer to the website.

E.11.1 Qualitative Analysis

Workshop environment

GraphEQA was provided with the following multiple choice question:

Question: *Where is the backpack?*
A. On the chair
B. On the table
Answer: **A**

The agent first takes a `<Goto_Object_node>(chair)` step in the environment after an initial rotate-in-place mapping operation to populate the scene graph, choosing to investigate a chair found in the environment. An explanation for this choice is provided by the VLM: “Objects like the chair or table might have the backpack, so checking close to these areas is essential.” The robot then begins navigating to the chair to determine if the backpack is located there. During execution of the trajectory toward the chair, GraphEQA leverages its task-relevant visual memory to score three images encountered on its way to the chair. These images are shown in Figure E.11.



Figure E.11: Task-relevant images selected by GraphEQA for the task “Where is the backpack?”.

After finishing the execution of this trajectory, GraphEQA answers the question with ‘On the chair’ and provides the following explanation of its answer: “The backpack is visually confirmed to be on the chair in the current image. The presence of the backpack on the chair makes it clear that the correct answer to the question is ‘On the chair’.”

Question: *Is the door to the lobby open?*
 A. Yes. B. No.
 Answer: **A**

In this particular experiment, the agent takes three planning steps, two of which constitute `<Goto_Object_node>(cabinet)` steps, and one `<Goto_Frontier_node>(frontier_id)`. For the step prior to the final action during which the agent answers the question correctly, the explanation of its current answer is “There is no direct observation of the lobby door in the current view or the scene graph. Since I’ve previously explored without finding the door, and considering I do not have visual confirmation, I cannot accurately determine if the door is open or closed.” after which the agent takes the frontier step to explore, finding the lobby door and correctly answering the question.

We ask the agent a second question regarding the location of a blue water bottle, along with three multiple choice answers.

Question: *Where is the blue water bottle?*
 A. On the table
 B. On the cabinet
 C. On the floor
 Answer: **B**

After exploring the environment with one `<Goto_Object_node>(cabinet)` step, see Figure 6.4, the agent successfully finds the water bottle and confirms its location, providing the following justification for its answer: “The image shows a cabinet with some objects on top, including a blue water bottle. There is also a computer monitor and various tools visible on the countertop.”

Office environment We ask the following question in an office setting.

Question: *Is my sweater on the blue couch?*
 A. Yes
 B. No Answer: **A**

The agent starts by taking a `<Goto_Object_node>(couch)` step, to explore the blue couch. The following VLM explanation of the object step clarifies GraphEQA is referring to the blue couch:

```
explanation_obj='I need to locate the blue couch before I can determine
if the sweater is on it or not.' object_id=<object_node_list.object_1: couch>
```

The low-level planner implementation on Hello Robot’s stretch does not plan the entire path to the blue couch, however, resulting in several more `<Goto_Object_node>(couch)` steps before answering the question confidently after 11 steps.

E.11.2 Additional Real-world Experiments in an Office Environment

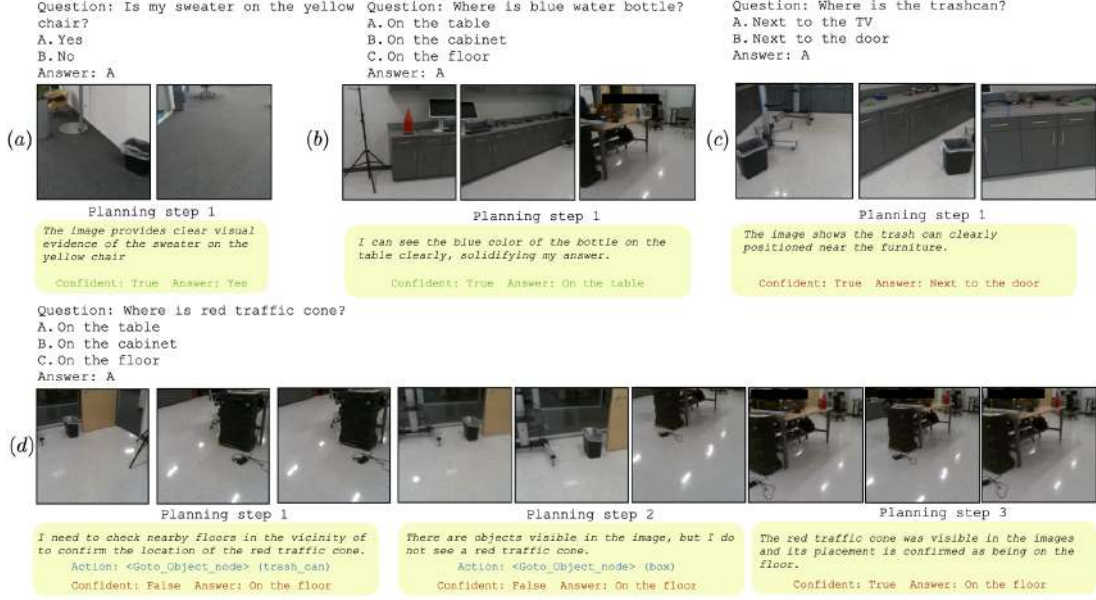


Figure E.12: Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in a workshop environment (b, c, d) and in an office (a). (c) and (d) are failure cases. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

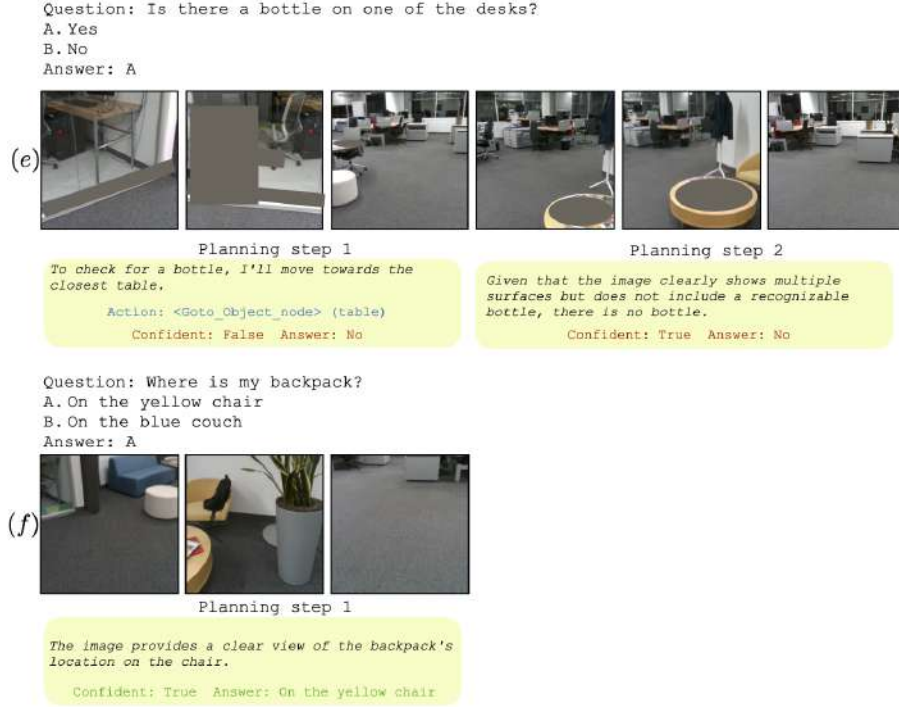


Figure E.13: Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in an office environment (e, f). (e) is a failure case. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

E.12 Qualitative results

E.12.1 Exploration Efficiency and Trajectory Generation

We illustrate the differences in exploration between GraphEQA and Explore-EQA through the following example. Figure E.14(a) shows the trajectory taken by an agent employing Explore-EQA in a scene in the HM3D dataset taking 30 VLM steps, while Figure E.14(b) shows the trajectory taken by an agent employing GraphEQA in that same scene taking a total of 5 VLM steps. We highlight here how Explore-EQA not only takes more steps, but that steps are often guided only by the semantic map constructed by images, leading to more exploration, while GraphEQA takes more structured steps in its environment, guided by the scene graph, to answer the question.

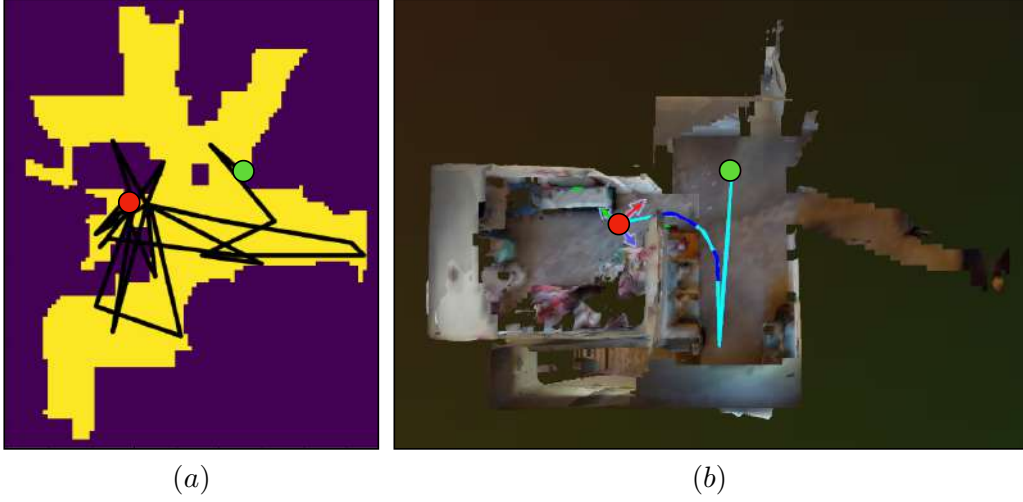


Figure E.14: Qualitative comparison of exploration trajectories of ExploreEQA vs GraphEQA. (a) A top down view of the map constructed by Explore-EQA illustrating explored areas for a scene in the HM3D dataset. The agent’s initial position is depicted in green and its final position in red. The trajectory of the agent is shown in black. (b) A top down view of scene reconstruction output by Hydra as GraphEQA explores the same HM3D scene used in the Explore-EQA map to the left. The agent’s initial position is again in green and its final position in red. The agent’s trajectory is shown in blue in this image. In this figure, we want to bring attention to the contrast between a typical trajectory taken by an agent employing GraphEQA vs. a typical trajectory taken by an agent employing Explore-EQA.

E.12.2 Choosing object-nodes and frontiers semantically

In E.12.2 we provide a qualitative examples of GraphEQAplanner choosing to go near a ‘couch’ to better answer a question about living room lights. This shows that the planner chooses task-relevant objects to guide exploration. In E.12.2 we provide a qualitative examples of GraphEQAplanner choosing task-relevant frontier nodes and object nodes to guide exploration.

Qualitative results - choosing task-relevant object nodes

```
Question:: Is the living room light on?
A. (Do not choose this option)
B. (Do not choose this option)
C. No
D. Yes.

Answer: D
At t=0::
Agent state: The agent is currently at node agent1 at position
[-2.928846836090088, -1.5354466438293457, 4.291640281677246] at room node:
room0 with name living room

Action: explanation room=I need to further investigate the living room to
possibly find indicators like light switches or bulbs. explanation obj=Going
near the blue couch might provide a better view of the surroundings, including
light fixtures. room id=room 0: 'living room' object id=object 13: 'couch'
Answer: explanation ans= I need to explore more areas related to lighting in

the living room before answering confidently. answer=Answer options A: '(Do
not choose this option)' explanation conf='The scene graph and current image
do not provide enough information on lighting. Going near specific objects
like the couch could help.' confidence level=0.2 is confident=False

At t=1::
Agent state: The agent is currently at node agent7 at position
[-2.2595670223236084, -2.6618852615356445, 4.291640281677246] at room node:
room 0 with name living room
Answer: explanation ans='The image clearly shows a light fixture on the

ceiling that appears to be on. This indicates that the living room light
is indeed on.' answer=D 'Yes' explanation conf='The light fixture being on is
visible in the image, providing clear evidence that the living room light is
on.' confidence level=0.9 is confident=True
```

Qualitative example of GraphEQAplanner choosing task-relevant object nodes for exploration.

Qualitative results: choosing task-relevant frontier nodes

Question:: Is the kitchen sink currently free of dishes?

A. (Do not choose this option)

B. (Do not choose this option)

C. Yes, it's empty

D. No, there are dishes.

Answer: C

At t=0::

Agent state: The agent is currently at node agent 1 at position [0.9115259647369385, -0.7087370157241821, -1.5000251531600952] at room node: room 0 with name bedroom

Action: explanation frontier="The kitchen typically contains objects like chairs, tables, etc. Frontier 4 is near objects that could indicate a transition to a dining or kitchen area, as it is near the couch and chair, which might lead to a kitchen. It's worth exploring this frontier to find the kitchen." frontier id= frontier 4

Answer: explanation ans="Currently, the kitchen is not visible in the scene. The scene graph and current image provide information about a bedroom, but not a kitchen. Therefore, I can't answer whether the kitchen sink is free of dishes." answer= A: '(Do not choose this option)' explanation conf="I haven't yet gathered any information on the kitchen or the sink from my current location in the bedroom, and no other regions of the building have been explored yet." confidence level=0.0 is confident=False

At t=1::

Agent state: The agent is currently at node agent 6 at position [2.5569498538970947, 2.228832721710205, -1.5000251531600952] at room node: room 0 with name living room

Action: explanation frontier="The frontier 2 is connected to objects like the table, cabinet, fridge, and plates, which might lead to a kitchen area or provide a clue to the sink's status." frontier id=frontier 2

Answer: explanation ans='Currently, there is no visual confirmation of a sink in the image or scene graph, and insufficient detail to determine if the kitchen sink is free of dishes.' answer=A: '(Do not choose this option)' explanation conf="The scene graph and current image give no direct view of a kitchen sink or its status, so it's uncertain whether it's free of dishes." confidence level=0.2 is confident=False

At t=2::

Agent state: The agent is currently at node agent 11 at position [2.6420648097991943, -0.10468387603759766, -1.5000251531600952] at room node: room 0 with name living room

Action: explanation room='To determine if the kitchen sink is free of dishes, I need to transition from the living room to the kitchen area, where the sink is likely located. Frontiers associated with kitchen-related objects can guide me there.' explanation obj="To check the sink, moving towards the 'plates' near object 41 might lead to the kitchen area." room id=room 0: 'living room' object id=object 41: 'plates'

:

⋮

Answer: explanation ans="Currently, I don't have enough information to directly answer if the kitchen sink is free of dishes, based on the image or scene graph. Further exploration is needed to

confirm the state of the sink." answer=A: '(Do not choose this option)' explanation conf="Given my current location and the items visible, I can't see the sink or its contents. I need to navigate closer to where I suspect the sink might be, based on kitchen-related objects like plates." confidence level=0.3 is confident=False

At t=3::

Agent state:The agent is currently at node agent 21 at position [1.7573769092559814, -0.35986196994781494, -1.5000251531600952] at room node: room 0 with name living room

Answer: explanation ans='The image shows the kitchen, and the sink area is visible. There

are no dishes in the sink, indicating it is empty.' answer=C: "Yes, it's empty" explanation conf='The image clearly shows the sink, and there are no dishes visible, providing clear evidence to confidently answer the question.' confidence level=1.0 is confident=True

Qualitative example of GraphEQApplanner choosing task-relevant frontier and object nodes for exploration.

Full VLM Prompt

Agent prompt: You are an excellent hierarchical graph planning agent. Your goal is to navigate an unseen environment to confidently answer a multiple-choice question about the environment. As you explore the environment, your sensors are building a scene graph representation (in json format) and you have access to that scene graph.

Scene graph description: A scene graph represents an indoor environment in a hierarchical tree structure consisting of nodes and edges/links. There are six types of nodes: building, rooms, visited areas, frontiers, objects, and agent in the environment. The tree structure is as follows: At the highest level 5 is a 'building' node. At level 4 are room nodes. There are links connecting the building node to each room node. At the lower level 3, are region and frontier nodes. 'region' node represent region of room that is already explored. Frontier nodes represent areas that are at the boundary of visited and unexplored areas. There are links from room nodes to corresponding region and frontier nodes depicting which room they are located in. At the lowest level 2 are object nodes and agent nodes. There is an edge from region node to each object node depicting which visited area of which room the object is located in. There are also links between frontier nodes and objects nodes, depicting the objects in the vicinity of a frontier node. Finally the agent node is where you are located in the environment. There is an edge between a region node and the agent node, depicting which visited area of which room the agent is located in.

Current state description: CURRENT STATE will give you the exact location of the agent in the scene graph by giving you the agent node id, location, room_id and room name.

General Description: Given the current state information, try to answer the question. Explain the reasoning for selecting the answer. Finally, report whether you are confident in answering the question. Explain the reasoning behind the confidence level of your answer. Rate your

level of confidence. Provide a value between 0 and 1; 0 for not confident at all and 1 for absolutely certain. Do not use just commonsense knowledge to decide confidence. Choose TRUE, if you have explored enough and are certain about answering the question correctly and no further exploration will help you answer the question better. Choose 'FALSE', if you are uncertain of the answer and should explore more to ground your answer in the current environment. Clarification: This is not your confidence in choosing the next action, but your confidence in answering the question correctly. If you are unable to answer the question with high confidence, and need more information to answer the question, then you can take two kinds of steps in the environment: Goto_object_node_step or Goto_frontier_node_step You also have to choose the next action, one which will enable you to answer the question better. Goto_object_node_step: Navigates near a certain object in the scene graph. Choose this action to get a good view of the region around this object, if you think going near this object will help you answer the question better. Important to note, the scene contains incomplete information about the environment (objects maybe missing, relationships might be unclear), so it is useful to go near relevant objects to get a better view to answer the question.

:

:

Use a scene graph as an imperfect guide to lead you to relevant regions to inspect. Choose the object in a hierarchical manner by first reasoning about which room you should goto to best answer the question, and then choose the specific object. Goto_frontier.node.step: If you think that using action ‘‘Goto_object.node.step’’ is not useful, in other words, if you think that going near any of the object nodes in the current scene graph will not provide you with any useful information to answer the question better, then choose this action. This action will navigate you to a frontier (unexplored) region of the environment and will provide you information about new objects/rooms not yet in the scene graph. It will expand the scene graph. Choose this frontier based on the objects connected this frontier, in other words, Goto the frontier near which you see objects that are useful for answering the question or seem useful as a good exploration direction. Explain reasoning for choosing this frontier, by listing the list of objects (<id> and <name>) connected to this frontier node via a link (refer to scene graph). While choosing either of the above actions, play close attention to ‘HISTORY’ especially the previous ‘Action’s to see if you have taken the same action at previous time steps. Avoid taking the same actions you have taken before. Describe the CURRENT IMAGE. Pay special attention to features that can help answer the question or select future actions. Describe the SCENE GRAPH. Pay special attention to features that can help answer the question or select future actions.’’

Prompt 1: The prompt used in the implementation of GraphEQA.

BIBLIOGRAPHY

- [1] S. Saxena, A. LaGrassa, and O. Kroemer, “Learning reactive and predictive differentiable controllers for switching linear dynamical models,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7563–7569, IEEE, 2021.
- [2] S. Saxena and O. Kroemer, “Dynamic inference on graphs using structured transition models,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10913–10920, IEEE, 2022.
- [3] S. Saxena, M. Sharma, and O. Kroemer, “Multi-resolution sensing for real-time control with vision-language models,” in *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.
- [4] S. Saxena, B. Buchanan, C. Paxton, B. Chen, N. Vaskevicius, L. Palmieri, J. Francis, and O. Kroemer, “Grapheqa: Using 3d semantic scene graphs for real-time embodied question answering,” *Conf. on Robot Learning (CoRL)*, 2025.
- [5] S. Saxena, O. Kroemer, and A. Bajcsy, “Vision-language guided safe dynamic manipulation via object-centric transformers.” https://saumyasaxena.github.io/assets/papers/safety_manipulation_2025_arxiv_v1.pdf, 2025. Under review.
- [6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, “ π 0: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550,” *arXiv preprint ARXIV.2410.24164*.
- [7] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, *et al.*, “Gemini robotics: Bringing ai into the physical world,” *arXiv preprint arXiv:2503.20020*, 2025.
- [8] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, “Rt-2: Vision-language-action models transfer web

- knowledge to robotic control,” in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.
- [9] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, 2018.
- [10] M. Janner, J. Fu, M. Zhang, and S. Levine, “When to trust your model: Model-based policy optimization,” in *Advances in Neural Information Processing Systems*, pp. 12519–12530, 2019.
- [11] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, “Towards learning hierarchical skills for multi-phase manipulation tasks,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1503–1510, IEEE, 2015.
- [12] P. Becker-Ehmck, J. Peters, and P. Van Der Smagt, “Switching linear dynamics for variational bayes filtering,” *arXiv preprint arXiv:1905.12434*, 2019.
- [13] S. W. Linderman, A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson, “Recurrent switching linear dynamical systems,” *arXiv preprint arXiv:1610.08466*, 2016.
- [14] J. J. Dabrowski, J. P. de Villiers, and C. Beyers, “Naïve bayes switching linear dynamical system: A model for dynamic system modelling, classification, and information fusion,” *Information Fusion*, vol. 42, pp. 75–101, 2018.
- [15] M. Toussaint and S. Vijayakumar, “Learning discontinuities with products-of-sigmoids for switching between local models,” in *Proceedings of the 22nd international conference on Machine Learning*, pp. 904–911, 2005.
- [16] G. Lee, Z. Marinho, A. M. Johnson, G. J. Gordon, S. S. Srinivasa, and M. T. Mason, “Unsupervised learning for nonlinear piecewise smooth hybrid systems,” *arXiv preprint arXiv:1710.00440*, 2017.
- [17] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable mpc for end-to-end planning and control,” in *Advances in Neural Information Processing Systems*, pp. 8289–8300, 2018.
- [18] P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Pérez, “Differentiable algorithm networks for composable robot learning,” *arXiv preprint arXiv:1905.11602*, 2019.
- [19] M. Toussaint, K. Allen, K. A. Smith, and J. B. Tenenbaum, “Differentiable physics and stable modes for tool-use and manipulation planning,” in *Robotics: Science and Systems*, 2018.

- [20] N. Sugimoto, J. Morimoto, S.-H. Hyon, and M. Kawato, “The emosaic model for humanoid robot control,” *Neural Networks*, vol. 29, pp. 8–19, 2012.
- [21] O. Kroemer, S. Niekum, and G. Konidaris, “A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms,” *arXiv preprint arXiv:1907.03146*, 2019.
- [22] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous deep q-learning with model-based acceleration,” in *International Conference on Machine Learning*, pp. 2829–2838, 2016.
- [23] N. Mishra, P. Abbeel, and I. Mordatch, “Prediction and control with temporal segment models,” *arXiv preprint arXiv:1703.04070*, 2017.
- [24] L. Sergey, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA*, pp. 26–30, 2015.
- [25] G. E. Dullerud and F. Paganini, *A course in robust control theory: a convex approach*, vol. 36. Springer Science & Business Media, 2013.
- [26] S. Pfrommer, M. Halm, and M. Posa, “Contactnets: Learning of discontinuous contact dynamics with smooth, implicit representations,” *arXiv preprint arXiv:2009.11193*, 2020.
- [27] N. Fazeli, R. Kolbert, R. Tedrake, and A. Rodriguez, “Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1437–1454, 2017.
- [28] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters, “Learning inverse dynamics models with contacts,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3186–3191, IEEE, 2015.
- [29] S. A. Khader, H. Yin, P. Falco, and D. Kragic, “Data-efficient model learning and prediction for contact-rich manipulation tasks,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4321–4328, 2020.
- [30] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, “Learning data-efficient rigid-body contact models: Case study of planar impact,” in *Conference on Robot Learning*, pp. 388–397, 2017.
- [31] S. Manschitz, J. Kober, M. Gienger, and J. Peters, “Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations,” *Robotics and Autonomous Systems*, vol. 74, pp. 97–107, 2015.

- [32] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, “Learning dexterous in-hand manipulation,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.
- [33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [34] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, “Q-prop: Sample-efficient policy gradient with an off-policy critic,” *arXiv preprint arXiv:1611.02247*, 2016.
- [35] K. Zhang, M. Sharma, J. Liang, and O. Kroemer, “A Modular Robotic Arm Control Stack for Research: Franka-Interface and FrankaPy,” *arXiv preprint arXiv:2011.02398*, 2020.
- [36] J. Wang and E. Olson, “Apriltag 2: Efficient and robust fiducial detection,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4193–4198, IEEE, 2016.
- [37] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, *et al.*, “Interaction networks for learning about objects, relations and physics,” *Advances in neural information processing systems*, vol. 29, 2016.
- [38] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [39] Y. Hoshen, “Vain: Attentional multi-agent predictive modeling,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [40] K. J. Åström and R. M. Murray, *Feedback systems, Chapter 8*. Princeton university press, 2010.
- [41] A. D. Dragan, “Robot planning with mathematical models of human state and action,” *arXiv preprint arXiv:1705.04226*, 2017.
- [42] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, “Learning visual predictive models of physics for playing billiards,” *arXiv preprint arXiv:1511.07404*, 2015.
- [43] A. Lerer, S. Gross, and R. Fergus, “Learning physical intuition of block towers by example,” in *International conference on machine learning*, pp. 430–438, PMLR, 2016.

- [44] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, “Learning to simulate complex physics with graph networks,” in *International Conference on Machine Learning*, pp. 8459–8468, PMLR, 2020.
- [45] Z. Weng, F. Paus, A. Varava, H. Yin, T. Asfour, and D. Kragic, “Graph-based task-specific prediction models for interactions between deformable and rigid objects,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5741–5748, IEEE, 2021.
- [46] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, “Visual interaction networks: Learning a physics simulator from video,” *Advances in neural information processing systems*, vol. 30, 2017.
- [47] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, “Graph networks as learnable physics engines for inference and control,” in *International Conference on Machine Learning*, pp. 4470–4479, PMLR, 2018.
- [48] T. Wang, R. Liao, J. Ba, and S. Fidler, “Nervenet: Learning structured policy with graph neural networks,” in *International conference on learning representations*, 2018.
- [49] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu, “Reasoning about physical interactions with object-oriented prediction and planning,” *arXiv preprint arXiv:1812.10972*, 2018.
- [50] Y. Li, A. Torralba, A. Anandkumar, D. Fox, and A. Garg, “Causal Discovery in Physical Systems from Videos,” *Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.
- [51] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” in *International Conference on Machine Learning*, pp. 2688–2697, PMLR, 2018.
- [52] F. Alet, E. Weng, T. Lozano-Pérez, and L. P. Kaelbling, “Neural relational inference with fast modular meta-learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [53] S. Van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions,” *arXiv preprint arXiv:1802.10353*, 2018.
- [54] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf, “Recurrent independent mechanisms,” *arXiv preprint arXiv:1909.10893*, 2019.

- [55] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, “Embed to control: A locally linear latent dynamics model for control from raw images,” *Advances in neural information processing systems*, vol. 28, 2015.
- [56] E. Banijamali, R. Shu, H. Bui, A. Ghodsi, *et al.*, “Robust locally-linear controllable embedding,” in *International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759, PMLR, 2018.
- [57] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, “Solar: Deep structured representations for model-based reinforcement learning,” in *International Conference on Machine Learning*, pp. 7444–7453, PMLR, 2019.
- [58] S. Levine and P. Abbeel, “Learning neural network policies with guided policy search under unknown dynamics,” *Advances in neural information processing systems*, vol. 27, 2014.
- [59] H. Yin, M. C. Welle, and D. Kragic, “Policy learning with embedded koopman optimal control,” 2021.
- [60] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.
- [61] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.
- [63] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022.
- [64] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, “Bc-z: Zero-shot task generalization with robotic imitation learning,” in *Conference on Robot Learning*, pp. 991–1002, PMLR, 2022.
- [65] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, “A generalist agent,” *arXiv preprint arXiv:2205.06175*, 2022.
- [66] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.

- [67] J. Li, R. Selvaraju, A. Gotmare, S. Joty, C. Xiong, and S. C. H. Hoi, “Align before fuse: Vision and language representation learning with momentum distillation,” *Advances in neural information processing systems*, vol. 34, pp. 9694–9705, 2021.
- [68] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, *et al.*, “Flamingo: a visual language model for few-shot learning,” *arXiv preprint arXiv:2204.14198*, 2022.
- [69] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, “Scaling up visual and vision-language representation learning with noisy text supervision,” in *International Conference on Machine Learning*, pp. 4904–4916, PMLR, 2021.
- [70] M. Shridhar, L. Manuelli, and D. Fox, “Cliport: What and where pathways for robotic manipulation,” in *Conference on Robot Learning*, pp. 894–906, PMLR, 2022.
- [71] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” *arXiv preprint arXiv:2209.05451*, 2022.
- [72] M. Sharma, C. Fantacci, Y. Zhou, S. Koppula, N. Heess, J. Scholz, and Y. Aytar, “Lossless adaptation of pretrained vision models for robotic manipulation,” in *The Eleventh International Conference on Learning Representations*.
- [73] M. Wortsman, G. Ilharco, J. W. Kim, M. Li, S. Kornblith, R. Roelofs, R. G. Lopes, H. Hajishirzi, A. Farhadi, H. Namkoong, *et al.*, “Robust fine-tuning of zero-shot models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022.
- [74] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, “Understanding natural language commands for robotic navigation and mobile manipulation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, pp. 1507–1514, 2011.
- [75] M. R. Walter, S. M. Hemachandra, B. S. Homberg, S. Tellex, and S. Teller, “Learning semantic maps from natural language descriptions,” *Robotics: Science and Systems*, 2013.
- [76] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox, “Learning from unscripted deictic gesture and language for human-robot interactions,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, 2014.
- [77] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, “Progprompt: Generating situated robot task plans using large language models,” *arXiv preprint arXiv:2209.11302*, 2022.

- [78] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, “Do as i can and not as i say: Grounding language in robotic affordances,” in *arXiv preprint arXiv:2204.01691*, 2022.
- [79] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, “Clip on wheels: Zero-shot object navigation as object localization and exploration,” *arXiv preprint arXiv:2203.10421*, 2022.
- [80] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” *arXiv preprint arXiv:2209.07753*, 2022.
- [81] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, “Text2motion: From natural language instructions to feasible plans,” *arXiv preprint arXiv:2303.12153*, 2023.
- [82] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela, “Flava: A foundational language and vision alignment model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15638–15650, 2022.
- [83] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, “R3m: A universal visual representation for robot manipulation,” *arXiv preprint arXiv:2203.12601*, 2022.
- [84] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, “The unsurprising effectiveness of pre-trained vision models for control,” *arXiv preprint arXiv:2203.03580*, 2022.
- [85] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang, “Vlmbench: A compositional benchmark for vision-and-language manipulation,” *arXiv preprint arXiv:2206.08522*, 2022.
- [86] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson, “Robotic skill acquisition via instruction augmentation with vision-language models,” *arXiv preprint arXiv:2211.11736*, 2022.
- [87] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, and K. Hausman, “Open-world object manipulation using pre-trained vision-language model,” in *arXiv preprint*, 2023.

- [88] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, “Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.
- [89] O. Mees, L. Hermann, and W. Burgard, “What matters in language conditioned robotic imitation learning over unstructured data,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11205–11212, 2022.
- [90] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
- [91] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” *arXiv preprint arXiv:1806.10293*, 2018.
- [92] A. X. Lee, C. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, C. Fantacci, J. E. Chen, A. Raju, R. Jeong, M. Neunert, A. Laurens, S. Saliceti, F. Casarini, M. Riedmiller, R. Hadsell, and F. Nori, “Beyond pick-and-place: Tackling robotic stacking of diverse shapes,” in *Conference on Robot Learning (CoRL)*, 2021.
- [93] B. H. Yoshimi and P. K. Allen, “Active, uncalibrated visual servoing,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 156–161, IEEE, 1994.
- [94] D. Kragic, H. I. Christensen, *et al.*, “Survey on Visual Servoing for Manipulation,” *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, p. 2002, 2002.
- [95] B. J. Nelson, J. D. Morrow, and P. K. Khosla, “Improved force control through visual servoing,” in *Proceedings of 1995 American Control Conference-ACC’95*, vol. 1, pp. 380–386, IEEE, 1995.
- [96] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards,” *arXiv preprint arXiv:1707.08817*, 2017.
- [97] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. Bekris, “Vision-driven compliant manipulation for reliable, high-precision assembly tasks,” *arXiv preprint arXiv:2106.14070*, 2021.

- [98] E. Johns, “Coarse-to-fine imitation learning: Robot manipulation from a single demonstration,” in *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 4613–4619, IEEE, 2021.
- [99] O. Spector and D. Di Castro, “Insertionnet-a scalable solution for insertion,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, 2021.
- [100] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, “Making sense of vision and touch: Self-supervised learning of multi-modal representations for contact-rich tasks,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8943–8950, IEEE, 2019.
- [101] N. Fazeli, M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, and A. Rodriguez, “See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion,” *Science Robotics*, vol. 4, no. 26, p. eaav3123, 2019.
- [102] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, “More than a feeling: Learning to grasp and regrasp using vision and touch,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.
- [103] Q. Li, O. Kroemer, Z. Su, F. F. Veiga, M. Kaboli, and H. J. Ritter, “A review of tactile information: Perception and action through touch,” *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1619–1634, 2020.
- [104] T. Narita and O. Kroemer, “Policy blending and recombination for multi-modal contact-rich tasks,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2721–2728, 2021.
- [105] S. Singh, F. M. Ramirez, J. Varley, A. Zeng, and V. Sindhwani, “Multiscale sensor fusion and continuous control with neural cdes,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10897–10904, IEEE, 2022.
- [106] K.-i. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous time recurrent neural networks,” *Neural networks*, vol. 6, no. 6, pp. 801–806, 1993.
- [107] P. Kidger, J. Morrill, J. Foster, and T. Lyons, “Neural controlled differential equations for irregular time series,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6696–6707, 2020.
- [108] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, and K. M. Lynch, “Dynamic in-hand sliding manipulation,” *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 778–795, 2017.

- [109] C. Mucchiani and M. Yim, “Dynamic grasping for object picking using passive zero-dof end-effectors,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3089–3096, 2021.
- [110] R. Shu and R. Hollis, “Momentum based whole-body optimal planning for a single-spherical-wheeled balancing mobile manipulator,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3221–3226, IEEE, 2021.
- [111] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, “Mdetr-modulated detection for end-to-end multi-modal understanding,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1780–1790, 2021.
- [112] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 213–229, Springer, 2020.
- [113] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [114] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
- [115] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, “Reinforcement learning with augmented data,” *Advances in neural information processing systems*, vol. 33, pp. 19884–19895, 2020.
- [116] N. Somavarapu, C.-Y. Ma, and Z. Kira, “Frustratingly simple domain generalization via image stylization,” *arXiv preprint arXiv:2006.11207*, 2020.
- [117] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, “Rlbench: The robot learning benchmark & learning environment,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.
- [118] U. Nagarajan, G. Kantor, and R. Hollis, “The ballbot: An omnidirectional balancing mobile robot,” *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, 2014.
- [119] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” in *Conference on robot learning*, pp. 1094–1100, PMLR, 2020.

- [120] Y. Zhou, S. Sonawani, M. Phielipp, S. Stepputtis, and H. B. Amor, “Modularity through attention: Efficient training and transfer of language-conditioned policies for robot manipulation,” *arXiv preprint arXiv:2212.04573*, 2022.
- [121] W. Yuan, S. Dong, and E. H. Adelson, “Gelsight: High-resolution robot tactile sensors for estimating geometry and force,” *Sensors*, vol. 17, no. 12, p. 2762, 2017.
- [122] A. Yamaguchi and C. G. Atkeson, “Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 1045–1051, IEEE, 2016.
- [123] K. Zhang, M. Sharma, M. Veloso, and O. Kroemer, “Leveraging multimodal haptic sensory data for robust cutting,” in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 409–416, IEEE, 2019.
- [124] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, “Safety and liveness guarantees through reach-avoid reinforcement learning,” *arXiv preprint arXiv:2112.12288*, 2021.
- [125] N. Hogan, “Impedance control: An approach to manipulation,” in *1984 American control conference*, pp. 304–313, IEEE, 1984.
- [126] F. J. Abu-Dakka and M. Saveriano, “Variable impedance control and learning—A review,” *Frontiers in Robotics and AI*, vol. 7, p. 590681, 2020.
- [127] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, “Task-space control of robot manipulators with null-space compliance,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2013.
- [128] A. De Luca and L. Ferrajoli, “Exploiting robot redundancy in collision detection and reaction,” in *2008 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3299–3305, IEEE, 2008.
- [129] Z. Li, M. Zamora, H. Zheng, and S. Coros, “Embracing safe contacts with contact-aware planning and control,” *arXiv preprint arXiv:2308.04323*, 2023.
- [130] X. Zhu, W. Lian, B. Yuan, C. D. Freeman, and M. Tomizuka, “Allowing safe contact in robotic goal-reaching: Planning and tracking in operational and null spaces,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8120–8126, IEEE, 2023.
- [131] L. Peternel, T. Petrič, and J. Babič, “Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface,” in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 1497–1502, IEEE, 2015.

- [132] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, “Force-based learning of variable impedance skills for robotic manipulation,” in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–9, IEEE, 2018.
- [133] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka, “Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning,” in *Conference on Robot Learning*, pp. 1621–1639, PMLR, 2023.
- [134] X. Zhu, S. Kang, and J. Chen, “A contact-safe reinforcement learning framework for contact-rich robot manipulation,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2476–2482, IEEE, 2022.
- [135] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, “Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks,” in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 1010–1017, IEEE, 2019.
- [136] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, “Hamilton-jacobi reachability: A brief overview and recent advances,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253, IEEE, 2017.
- [137] K. Margellos and J. Lygeros, “Hamilton–jacobi formulation for reach–avoid differential games,” *IEEE Transactions on automatic control*, vol. 56, no. 8, pp. 1849–1861, 2011.
- [138] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, “A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games,” *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.
- [139] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 4. Athena scientific, 2012.
- [140] B. D. O. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.
- [141] J. Darbon and S. Osher, “Algorithms for overcoming the curse of dimensionality for certain hamilton–jacobi equations arising in control theory and elsewhere,” *Research in the Mathematical Sciences*, vol. 3, no. 1, p. 19, 2016.
- [142] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, “Bridging hamilton-jacobi safety analysis and reinforcement learning,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8550–8556, IEEE, 2019.

- [143] A. K. Akametalu, S. Ghosh, J. F. Fisac, V. Rubies-Royo, and C. J. Tomlin, “A minimum discounted reward hamilton-jacobi formulation for computing reachable sets,” *IEEE Transactions on Automatic Control*, 2023.
- [144] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine learning*, vol. 3, pp. 9–44, 1988.
- [145] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.
- [146] J. Varley, S. Singh, D. Jain, K. Choromanski, A. Zeng, S. B. R. Chowdhury, A. Dubey, and V. Sindhwani, “Embodied ai with two arms: Zero-shot learning, safety and modularity,” *arXiv preprint arXiv:2404.03570*, 2024.
- [147] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, IEEE, 2023.
- [148] Z. Yang, S. S. Raman, A. Shah, and S. Tellex, “Plug in the safety chip: Enforcing constraints for llm-driven robot agents,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14435–14442, IEEE, 2024.
- [149] H. Wang, N. Chin, G. Gonzalez-Pumariega, X. Sun, N. Sunkara, M. A. Pace, J. Bohg, and S. Choudhury, “Apricot: Active preference learning and constraint-aware task planning with llms,” *arXiv preprint arXiv:2410.19656*, 2024.
- [150] N. Kumar, F. Ramos, D. Fox, and C. R. Garrett, “Open-world task and motion planning via vision-language model inferred constraints,” *arXiv preprint arXiv:2411.08253*, 2024.
- [151] A. Curtis, N. Kumar, J. Cao, T. Lozano-Pérez, and L. P. Kaelbling, “Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction,” *arXiv preprint arXiv:2406.05572*, 2024.
- [152] L. Brunke, Y. Zhang, R. Römer, J. Naimier, N. Staykov, S. Zhou, and A. P. Schoellig, “Semantically safe robot manipulation: From semantic scene understanding to motion safeguards,” *arXiv preprint arXiv:2410.15185*, 2024.
- [153] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
- [154] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

- [155] T. Lillicrap, “Continuous control with deep reinforcement learning,” *International Conference on Learning Representations*, 2016.
- [156] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, “Google scanned objects: A high-quality dataset of 3d scanned household items,” in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2553–2560, IEEE, 2022.
- [157] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, “Google scanned objects: A high-quality dataset of 3d scanned household items,” 2022.
- [158] K. Zakka, “Scanned Objects MuJoCo Models,” 7 2022.
- [159] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, “robosuite: A modular simulation framework and benchmark for robot learning,” in *arXiv preprint arXiv:2009.12293*, 2020.
- [160] e. a. OpenAI Team, “Gpt-4o system card,” 2024.
- [161] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman, “TAPIR: Tracking any point with per-frame initialization and temporal refinement,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10061–10072, 2023.
- [162] C. Doersch, A. Gupta, L. Markeeva, A. Recasens, L. Smaira, Y. Aytar, J. Carreira, A. Zisserman, and Y. Yang, “TAP-vid: A benchmark for tracking any point in a video,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 13610–13626, 2022.
- [163] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, “Embodied question answering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–10, 2018.
- [164] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, “Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning,” 2023.
- [165] A. Z. Ren, J. Clark, A. Dixit, M. Itkina, A. Majumdar, and D. Sadigh, “Explore until confident: Efficient exploration for embodied question answering,” 2024.
- [166] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, “Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning,” 2023.

- [167] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. MartÅn-MartÅn, “Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation,” 2024.
- [168] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang, “Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation,” 2024.
- [169] Q. Xie, S. Y. Min, T. Zhang, K. Xu, A. Bajaj, R. Salakhutdinov, M. Johnson-Roberson, and Y. Bisk, “Embodied-rag: General non-parametric embodied memory for retrieval and generation,” 2024.
- [170] Z. Xu, H.-T. L. Chiang, Z. Fu, M. G. Jacob, T. Zhang, T.-W. E. Lee, W. Yu, C. Schenck, D. Rendleman, D. Shah, *et al.*, “Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs,” in *8th Annual Conference on Robot Learning*, 2024.
- [171] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, *et al.*, “Openeqa: Embodied question answering in the era of foundation models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16488–16498, 2024.
- [172] R. Xu, Z. Huang, T. Wang, Y. Chen, J. Pang, and D. Lin, “Vlm-grounder: A vlm agent for zero-shot 3d visual grounding,” *arXiv preprint arXiv:2410.13860*, 2024.
- [173] A. Werby, C. Huang, M. BÅijchner, A. Valada, and W. Burgard, “Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation,” in *Robotics: Science and Systems XX*, RSS2024, Robotics: Science and Systems Foundation, July 2024.
- [174] N. Hughes, Y. Chang, and L. Carlone, “Hydra: A real-time spatial perception system for 3d scene graph construction and optimization,” 2022.
- [175] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, K. Yadav, Q. Li, B. Newman, M. Sharma, V. Berges, S. Zhang, P. Agrawal, Y. Bisk, D. Batra, M. Kalakrishnan, F. Meier, C. Paxton, S. Sax, and A. Rajeswaran, “Openeqa: Embodied question answering in the era of foundation models,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [176] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, *et al.*, “Habitat-matterport 3d semantics dataset,” *arXiv preprint arXiv:2210.05633*, 2022.

- [177] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, “3d scene graph: A structure for unified semantics, 3d space, and camera,” 2019.
- [178] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, “3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents,” *IEEE Transactions on Cybernetics*, vol. 50, p. 4921–4933, Dec. 2020.
- [179] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, “Kimera: from slam to spatial perception with 3d dynamic scene graphs,” 2021.
- [180] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone, “Clio: Real-time task-driven open-set 3d scene graphs,” *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8921–8928, 2024.
- [181] J. Loo, Z. Wu, and D. Hsu, “Open scene graphs for open world object-goal navigation,” 2024.
- [182] X. Zhou, R. Girdhar, A. Joulin, P. KrÄhenbÄijhl, and I. Misra, “Detecting twenty-thousand classes using image-level supervision,” 2022.
- [183] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, “Sigmoid loss for language image pre-training,” 2023.
- [184] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh, “Prismatic vlms: Investigating the design space of visually-conditioned language models,” *arXiv preprint arXiv:2402.07865*, 2024.
- [185] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, “Habitat 2.0: Training home assistants to rearrange their habitat,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [186] Y. Wang and E. Johns, “One-shot dual-arm imitation learning,” *arXiv preprint arXiv:2503.06831*, 2025.
- [187] M. Hong, A. Liang, K. Kim, H. Rajaprakash, J. Thomason, E. Bıyık, and J. Zhang, “Hand me the data: Fast robot adaptation via hand path retrieval,” *arXiv preprint arXiv:2505.20455*, 2025.
- [188] A. Bajcsy, D. P. Losey, M. K. O’Malley, and A. D. Dragan, “Learning from physical human corrections, one feature at a time,” in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 141–149, 2018.

Bibliography

- [189] A. N. Angelopoulos and S. Bates, “A gentle introduction to conformal prediction and distribution-free uncertainty quantification,” *arXiv preprint arXiv:2107.07511*, 2021.
- [190] J. Zhang and K. Cho, “Query-efficient imitation learning for end-to-end autonomous driving,” *arXiv preprint arXiv:1605.06450*, 2016.
- [191] R. Bellman, R. E. Kalaba, *et al.*, *Dynamic programming and modern control theory*, vol. 81. Citeseer, 1965.
- [192] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra, “Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.