Task And Motion Planning for Mobile Manipulation

Kim Tien Ly

St Hugh's College
University of Oxford

A thesis submitted for the degree of Doctor of Philosophy

Michaelmas 2024

Abstract

Task and Motion Planning (TAMP) aims to integrate high-level symbolic reasoning with low-level geometric reasoning, creating extensive plans grounded in actionable commands. By bridging decision-making and physical action, TAMP enables robots to perform complex, multi-step tasks.

This thesis contributes solutions to the TAMP problem for mobile manipulation through three distinct projects, which are designed to exploit unique challenges of TAMP. The frameworks address embodied intelligence through investigation across sampling, optimization, and learning methods.

First, a hierarchical optimization-based TAMP structure is introduced for coverage planning in legged manipulators, enabling efficient and robust execution with a dexterous task. Second, an approach combining sampling-based methods with optimal TAMP principles is developed, constructing a reachability graph for efficient state-space exploration. Finally, an LLM-based planner is designed to enable natural language interaction, lightweight adaptability, and failure recovery in domestic environments. These systems are validated via different robot deployments in industrial and domestic environments, aiming at robust and reliable robot autonomy.

In addition to solving TAMP, our frameworks design end-to-end solutions by processing perception input to the central planner, further contributing towards the autonomy and robustness of robotics systems.

Task And Motion Planning for Mobile Manipulation

Kim Tien Ly

St Hugh's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2024

Declaration

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Kim Tien Ly
St Hugh's College

Acknowledgements

First and foremost, I would like to extend my deepest gratitude to my supervisor, Prof. Ioannis Havoutis, for his unwavering support, guidance, and encouragement throughout my DPhil journey. His profound knowledge, insightful feedback, and constant availability have been instrumental in shaping this research. Professor Ioannis's mentorship has not only enhanced my academic skills but also inspired me to strive for excellence and resilience in the face of challenges. I am deeply thankful for his patience, understanding, and dedication, which have been pivotal in the successful completion of my thesis.

I am immensely grateful to Dr. Wolfgang Merkt for his invaluable guidance and support during the initial stages of my DPhil journey. His expertise, technical skills, thoughtful patience, and encouragement played a crucial role in helping me overcome early challenges. Although he is no longer at the Oxford Robotics Institute (ORI), he continues to provide support and guidance in shaping my career path, for which I am sincerely thankful.

I would also send a special thank you to all of my collaborators and DRS members for all the enriching discussions, support, and camaraderie throughout this journey. Their expertise and dedication have significantly enriched the quality and scope of my work, and collaborating with such talented and committed individuals has been a truly rewarding experience.

My sincere appreciation goes to the members of my thesis committee, Professor Marc Toussaint and Dr. Bruno Lacerda, for agreeing to examine my PhD viva. I also thank Prof. Nick Hawes, Prof. Ingmar Posner and Dr. Bruno Lacerda for assessing my work in the Transfer and Confirmation of Status, and giving me valuable comments to move forward with my research.

I must acknowledge the ORI for providing the essential resources and fostering a stimulating academic environment that has been fundamental to my research journey. I am grateful to the engineering team, especially Tobit and Benoit, for their excellent technical expertise in addressing robotics-related issues and ensuring the smooth progress of my work. I also deeply appreciate the admin team for answering my logistical questions and for their efforts in organizing bonding activities that helped build a strong sense of community within ORI. Their combined contributions have made my time at ORI both productive and memorable.

My time with the student robotics team, ORIon, has been an incredibly enriching experience, blending fun and learning as we worked with robots. It provided me with hands-on exposure to real robot systems, significantly enhancing my practical skills and understanding of robotics. Beyond the technical lessons, ORIon became a platform where I formed lasting friendships and shared memorable moments. I am deeply thankful to the team members for the insightful lessons, engaging conversations, and the camaraderie we built along the way. Together, we embarked on exciting journeys to RoboCup 2022 and 2023, creating unforgettable memories both on and off the field. These experiences have been a highlight of my academic journey, and I will always cherish the time I spent as part of this inspiring team.

Especially, I would like to thank the Vingroup Science and Technology Scholarship Program for their financial support throughout my DPhil studies. This scholarship has provided me with the resources and peace of mind necessary to focus entirely on my research and academic pursuits. Their commitment to fostering academic excellence and research innovation has been a source of great motivation and inspiration throughout my journey. In addition, I appreciate the travel financial support for the summer school from ETH Zurich, and for my conference trips from Prof. Ioannis, IEEE bodies, Department of Engineering Science, and St Hugh's College.

I am deeply grateful to my family for their unconditional love, support, and endless encouragement throughout this journey. To my parents, thank you for your boundless patience, guidance, and belief in me; your sacrifices and values have been my constant source of inspiration. To my partner, your understanding, reassurance, and committed presence have been my anchor during both the challenging and rewarding moments. To my friends, thank you for your companionship, kindness, and the laughter that brightened even the toughest days. Together, you have been my greatest strength, and I am forever thankful for having you by my side.

This thesis represents a significant milestone in both my academic journey and personal development. It encapsulates years of learning, challenges, and growth that have shaped me as a researcher and an individual. I am profoundly grateful to everyone who has contributed to this journey—whether through guidance, collaboration, mental or financial support. Their encouragement and belief in my potential have been invaluable, and this accomplishment would not have been possible without them. As I close this chapter, I carry forward not only the knowledge gained but also the cherished memories and lessons learned along the way.

Abstract

Task and Motion Planning (TAMP) aims to integrate high-level symbolic reasoning with low-level geometric reasoning, creating extensive plans grounded in actionable commands. By bridging decision-making and physical action, TAMP enables robots to perform complex, multi-step tasks.

This thesis contributes solutions to the TAMP problem for mobile manipulation through three distinct projects, which are designed to exploit unique challenges of TAMP. The frameworks address embodied intelligence through investigation across sampling, optimization, and learning methods.

First, a hierarchical optimization-based TAMP structure is introduced for coverage planning in legged manipulators, enabling efficient and robust execution with a dexterous task. Second, an approach combining sampling-based methods with optimal TAMP principles is developed, constructing a reachability graph for efficient state-space exploration. Finally, an LLM-based planner is designed to enable natural language interaction, lightweight adaptability, and failure recovery in domestic environments. These systems are validated via different robot deployments in industrial and domestic environments, aiming at robust and reliable robot autonomy.

In addition to solving TAMP, our frameworks design end-to-end solutions by processing perception input to the central planner, further contributing towards the autonomy and robustness of robotics systems.

Contents

List of Figures

List of Abbreviations

AI . . . . . Artificial Intelligence.

CASE . . . . . International Conference on Automation Science and Engineering.

CP . . . . . Constraint programming.

DoF . . . . . Degrees of Freedom.

DPhil . . . . . Doctor of Philosophy.

DRS . . . . . DRS Dynamic Robot Systems.

EE . . . . . End-Effector.

FOL . . . . . First-order logic.

HITL . . . . . Human In The Loop.

HSR . . . . . Toyota Human Support Robot.

ICRA . . . . . International Conference on Robotics and Automation.

IEEE . . . . . IEEE Institute of Electrical and Electronics Engineers.

IK . . . . . Inverse Kinematics.

IRM . . . . . Inverse Reachability Map.

IROS . . . . . International Conference on Intelligent Robots and Systems.

LLM . . . . . Large Language Models.

LGP . . . . . Logic-Geometric Programming.

MCTS . . . . . Monte Carlo Tree Search.

MILP . . . . . Mixed-integer linear programming

MIP . . . . . Mixed-integer programming.

PDDL . . . . . Planning Domain Definition Language.

PR2 . . . . . Personal Robot 2.

PRM . . . . . Probabilistic Roadmap.

RANSAC . . . Random Sample Consensus.

RGB-D . . . . Red Green Blue Depth.

RL . . . . . Reinforcement Learning.

SOTA . . . . . State-of-the-art.

STRIPS . . . . STanford Research Institute Problem Solver.

TAMP . . . . . Task And Motion Planning.

TSDF . . . . . Truncated Signed Distance Function.

VLM . . . . . Vision-Language Models.

1 Introduction

Contents

1.1 Motivation

The field of robotics has seen significant advancements in recent years, with mobile manipulation emerging as a critical area of research. Mobile manipulators, which combine mobility with the ability to manipulate objects, are increasingly used in diverse applications, from industrial automation to service robotics. The key research to enable these robots to perform complex tasks autonomously and robustly lies in Task and Motion Planning (TAMP). TAMP problems address the integration of task planning and motion planning, which specifies the process of end-to-end decision-making that takes a goal as input and outputs a sequence of robot configurations to complete the specified task. Hence, TAMP planners need to accommodate both task

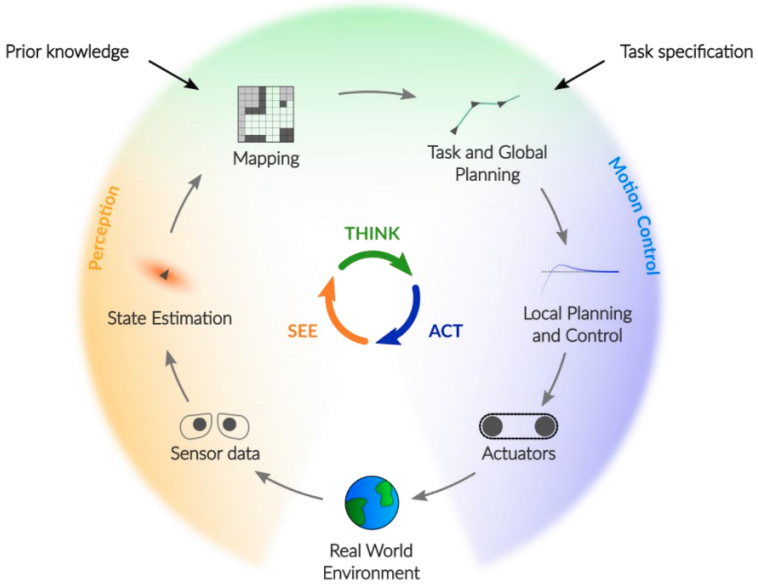


Figure 1.1: Robotic see-think-act cycle.

goals and low-level motion requirements, solving them either independently or jointly.

Figure 1.1 describes a typical see-think-act cycle in robot systems. In this layout, task planning refers to the 'think' feature, and motion planning and control denote the 'act' function. Let's take a kitchen manipulation task - getting a cup from the cupboard - as an example. If we know the steps that should be taken to finish the task (open cupboard, pick cup, close cupboard, place cup on the table), motion planning can take this sequence and produce robot joint configurations to fulfil the task. However, if we only specify that we need a cup on the table, task planning should also generate the desired sequence of actions. Our planner might also have to understand if we want a particular cup from the cupboard, and if other cups need to be moved first, in order to reach the desired one.

TAMP research presents numerous challenges due to its hybrid and complex nature. Even in static and known environments, developing a robust TAMP model capable of generating accurate sequences of robot configurations for multi-purpose real-world applications remains an open research question. As an integrated research field, TAMP encompasses problems that integrate planning both task and motion levels. A study by Mansouri et al. [1] identified five open questions that need to be addressed in TAMP design: (1) level abstractions, (2) joint reasoning of symbolic

and continuous knowledge, (3) integration of machine learning models with TAMP, (4) consistent online decision-making, and (5) issues related to uncertain perception. The authors also suggest that a one-size-fits-all solution is unlikely to exist.

TAMP has gained significant interest over the past few years [2] as researchers strive to address the above challenges. The advancement in various relevant technologies, ranging from image processing and language understanding to robot navigation and manipulation, has been instrumental in this pursuit. Research in TAMP generally aims to effectively integrate artificial intelligence techniques in task planning with advancements in robotics for motion planning, thereby enhancing the capabilities of automated systems to plan complex tasks efficiently.

1.2 Objective

This thesis aims to address the aforementioned gaps by exploring three distinct yet complementary approaches to TAMP for mobile manipulation. We aim to develop novel planners to solve end-to-end task and motion planning, focusing on the correlation between robot motion constraints and task-level requirements. Our goal is to achieve automated, robust, reliable, and ideally optimal solutions in both industrial and domestic applications.

The first objective is to investigate optimization-based methods in TAMP. This approach structures the planning process into manageable optimization layers, which obtain time and energy efficiency and effectiveness in both task and motion levels. The investigated task in this work, multi-surface coverage planning, specifically requires maintaining dexterity on legged manipulators.

In TAMP, the lack of robust integration between high-level task planning and low-level motion planning can result in inefficiencies and infeasible plans when the task planner is unaware of the geometric constraints at the motion level. To avoid this, we propose a sampling-based reachability graph where kinematic constraints affect high-level decisions. This method bridges the gap between high-level task planning and low-level motion constraints, ensuring that the generated task plans are both feasible and optimized within the environment. The integration

of geometric feedback into the optimization process can inform and refine the hierarchical planning system, creating a more cohesive and integrated planning framework, hence, reducing replanning efforts. We implemented this system in domestic settings with general tasks.

The recent advent of AI, particularly large language models (LLMs), offers new opportunities for task planning, but there is a need to explore how these models should be adjusted for embodied intelligence. The third objective is to explore the use of LLMs for TAMP, reducing the reliance on formal planning language and integrating human-in-the-loop. The project leverages the capabilities of AI to generate task plans that inherently respect the underlying kinematic requirements while recovering from failures with human guidance. In this work, we address TAMP solution's reactivity during deployment with end users.

Finally, sim-to-real has been raised as a typical issue deploying research work in real-world settings. Therefore, in our projects, we validated the algorithms on physical high degree-of-freedom (DoFs) mobile manipulators, verifying the contributions and aiming at realistic and practical applications.

1.3 Contributions

The main contribution of this thesis is the development of a set of TAMP planners for mobile manipulation. The methods address embodied intelligence where robots can reason around task requirements while understanding and respecting their kinematic specifications.

Specifically, the contributions are:

- Novel approaches to solve TAMP on legged and wheeled mobile manipulators.
- Robust and optimal TAMP solutions that respect reachability and kinematic constraints on high-DoF robots.
- Interactive and interpretable method taking advantage of textual understanding capability of LLMs.

Figure 1.2: Deployment of our TAMP solutions on legged and wheeled mobile manipulators, targeting both industrial and domestic applications.

- Real robots deployment showing robustness of the proposed TAMP solvers.

1.4 Publications

Shown below is a summary of the papers accepted and submitted as part of this degree. There is also a second-author publication listed.

1.4.1 First-author Papers

Kim Tien Ly, Matthew Munks, Wolfgang Merkt, and Ioannis Havoutis (2023). Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring. *Proc. IEEE International Conference on Automation Science and Engineering (CASE).*

Kim Tien Ly, Valeriy Semenov, Mattia Risiglione, Wolfgang Merkt, Ioannis Havoutis (2024). R-LGP: A Reachability-guided Logic-geometric Programming

Framework for Optimal Task and Motion Planning on Mobile Manipulators. *Proc. IEEE International Conference on Robotics and Automation (ICRA).*

Kim Tien Ly, Kai Lu, Ioannis Havoutis (2024). InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy. *Under Review.*

Workshop paper: **Kim Tien Ly**, Valeriy Semenov, Mattia Risiglione, Wolfgang Merkt, Ioannis Havoutis (2024). Towards a reachability-guided TAMP framework for mobile manipulation. *The 5th UK Robot Manipulation Workshop.*

1.4.2 Co-authored Papers

Kai Lu, **Kim Tien Ly**, William Hebberd, Kaichen Zhou, Ioannis Havoutis, Andrew Markham (2024). Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Mattia Risiglione, Abdelrahman Abdalla, Victor Barasuol, **Kim Tien Ly**, Ioannis Havoutis and Claudio Semini (2024). RAKOMO: Reachability-Aware K-Order Markov Path Optimization for Quadrupedal Loco-Manipulation. *Under Review.*

1.5 Thesis Outline

This thesis is presented in the integrated format of the University of Oxford. Chapters 4-6 consist of peer-reviewed publications, each one accompanied by additional discussion as well as a Statement of Authorship declaring the author's contribution to each work.

The remainder of the thesis is structured as follows:

- **Chapter 2 – Background**: Presentation of the main definitions, theory, and methods used in the thesis.

- **Chapter 3 - Related Work**: Review of the relevant literature on task and motion planning.

- **Chapter 4 - Hierarchical Optimization-based TAMP**: A hierarchical system to automatically and optimally deal with end-to-end planning on mobile manipulation in remote sites.
- **Chapter 5 - Reachability-guided Optimal TAMP**: An approach to provide high-level planner of the low-level constraint to avoid replanning in TAMP.
- **Chapter 6 - Interactive Lightweight Robotics Planner**: An interactive solution to onboard LLM-based planner with interpretable recovery behaviours.
- **Chapter 7 - Conclusion**: Discussion on the achievements and limitations of this work, lessons learned, and future directions for the field.

2 Background

Contents

This chapter introduces important background concepts and principles for the remainder of the thesis and will be referred to in later chapters.

2.1 Task and Motion Planning (TAMP)

2.1.1 TAMP Overview

TAMP is a fundamental problem in robotics that involves generating a sequence of actions and corresponding continuous movements, accounting for both discrete task constraints and continuous motion requirements. The key challenge in TAMP is the integration of *task planning* and *motion planning*.

The problem can be viewed as a search over two interdependent spaces:

- Task space: A discrete space representing symbolic actions, such as "pick", "place", or "move".
- Motion space: A continuous space defined by the robot's configuration space $\mathcal{C}$, including all its possible positions and orientations.

Formally, TAMP can be defined as follows:

Find (π, ξ) such that:

- $\pi = [a_0, a_1, a_2, \ldots, a_n]$ is a sequence of symbolic actions, where each $a_i \in \mathcal{A}$ is an action from a discrete action space $\mathcal{A}$. π operates under the state transition $s_i = \text{succ}(a_i, s_{i-1})$, where $s_n = s_g$.
- $\xi = [\tau_0, \tau_1, \tau_2, \ldots, \tau_n]$ is a sequence of motion trajectories to complete the symbolic sequence π, where each τ_i is a feasible trajectory in the robot's configuration space $\mathcal{C}$ to execute action a_i.

Communication Between Task Planner and Motion Planner

The high-level task planner and the low-level motion planner communicate through the sequence of actions π. Assuming there are k_i timesteps for each action a_i, the motion planner computes a trajectory $\tau_i = [\mathbf{q}_{i0}, \mathbf{q}_{i1}, \ldots, \mathbf{q}_{ik_i}]$ in the configuration space $\mathcal{C}$, where $\mathbf{q}_{ij}$ denotes the robot's configuration at a specific time step j within the trajectory.

To ensure motion feasibility when chaining actions in a high-level strategy, it is required that the end configuration of one trajectory matches the start configuration of the next. This can be formulated as:

$$\mathbf{q}_{i0} = \mathbf{q}_{(i-1)k_i},$$

where $\mathbf{q}_{i0}$ is the start configuration for action a_i, and $\mathbf{q}_{ik_i}$ is the end configuration for action a_i. The start configuration $\mathbf{q}_{i0}$ of action a_i equal to the final configuration $\mathbf{q}_{(i-1)k_i}$ of the previous action a_{i-1} ensures smooth transitions between actions.

While the high-level task planner determines the action sequence π, the low-level motion planner ensures that for each action a_i, there exists a corresponding feasible trajectory τ_i in the robot's configuration space. The motion trajectories are computed subject to kinematic and geometric equality and inequality constraints, allowing the execution of TAMP results. Hierarchical structure introduces straightforward communication between levels, where the symbolic actions condition the motion trajectories.

2.1.2 Hierarchical Structure

A common approach to solving TAMP problems is to use hierarchical decomposition. This method splits the problem into levels, where high-level task planning generates abstract plans, and low-level motion planning solves the actual trajectories for each task. Hierarchical decomposition addresses the complexity of TAMP by breaking it down into two layers:

- High-level task planner: At the high level, a symbolic task planner generates a sequence of abstract actions, such as "pick object" or "move to target". These actions are defined in a discrete space and are often task-specific.

- Low-level motion planner: The low-level motion planner solves the motion trajectories in the robot's configuration space. It ensures that each action can be executed safely and respects the robot's constraints.

Figure 2.1 illustrates the typical hierarchical planning in TAMP, where the high-level and low-level planners solve the task and motion problems separately. In general, the task planner receives the goal and plan the action sequence $\pi = [a_0, a_1, a_2, \ldots, a_n]$ to complete the task. Meanwhile, the motion planner takes in

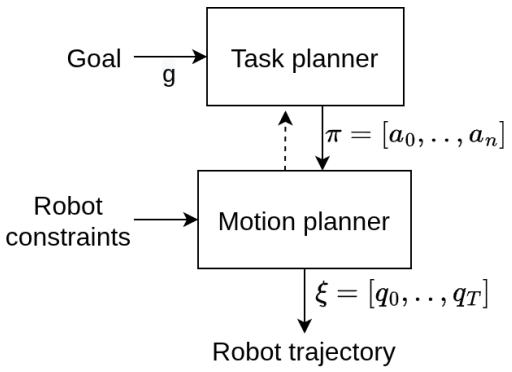


Figure 2.1: Hierarchical structure in TAMP

the action sequences with their task-specific constraint, and plan the full trajectory in $\xi = [\tau_0, \tau_1, \tau_2, \ldots, \tau_n] = [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T]$ respecting the robot constraints. In order to ensure executability, it is common to include a closed loop to check for motion planning success and re-plan if necessary.

2.1.3 Optimization Framework

Optimization-based approaches, such as Logic-Geometric Programming (LGP) [3], unify task planning and motion planning as a single optimization problem, where task-level constraints and motion-level constraints are solved simultaneously. This section provides the background knowledge of LGP, which was used in our thesis.

LGP formulates TAMP as an optimization problem that covers both discrete action sequence and continuous joints trajectory:

$$
\begin{aligned}
\text{minimize} \quad & \int_0^T c(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))\, dt + \psi(\mathbf{q}(T) \\
\text{subject to} \quad & s_i = \text{succ}(a_i, s_{i-1}), \quad \forall i = 1, \ldots, n, \\
& s_n \vDash g, \\
& h_{\text{switch}}(\mathbf{q}(t_i) \mid a_i, s_{i-1}) = 0, \quad \forall i = 1, \ldots, n, \\
& g_{\text{switch}}(\mathbf{q}(t_i) \mid a_i, s_{i-1}) \leq 0, \quad \forall i = 1, \ldots, n, \\
& h_{\text{path}}(\mathbf{q}(t, \dot{\mathbf{q}}(t) \mid s_i(t)) = 0, \quad \forall t \in [0, T], \\
& g_{\text{path}}(\mathbf{q}(t, \dot{\mathbf{q}}(t) \mid s_i(t)) \leq 0, \quad \forall t \in [0, T],
\end{aligned}
$$

where:

- $\mathbf{q}(t)$ represents the configurations of the robot over time.

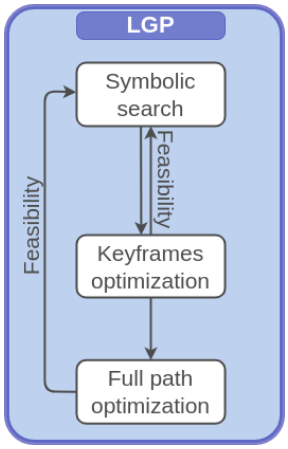


Figure 2.2: LGP structure

- c denotes the cost objective function.
- $\psi(\mathbf{q}(T)$ is the evaluation function for the final configuration of the system.
- s_i and a_i represent the symbolic states and actions respectively.
- g_{switch} and h_{switch} are geometric constraints related to action transitions.
- g_{path} and h_{path} are continuous path constraints.

Fig. 2.2 visualize the process of solving TAMP in LGP. The LGP solver employs a three-tiered approximation strategy to address the challenges posed by the integration of symbolic and geometric planning domains effectively. At the first level, the symbolic search quickly evaluate the end space, assessing the action transition to generate the symbolic sequence $\pi = [a_0, a_1, ..., a_n]$ to complete the given goal g. The mid-level focuses on optimizing jointly over all keyframe configurations $\mathbf{q}(t), t = t_0, t_1, ..., t_n$ conditional to π, verifying the feasibility of the action sequence. The keyframes essentially illustrate manipulation actions in the symbolic sequence π, in particular, where the robot's interaction with the environment changes, such as grasping or releasing an object. For example, with a_0 as *grasp* action, the mid-level optimize the configuration over the grasp pose at $t = t_0$, while trying to accommodate costs that arises at later timesteps. Lastly, the full path optimization layer solves the entire trajectory, ensuring a smooth trajectory ξ over $[0, T]$, conditional to the given action sequence π. In this sense, instead of planning individual trajectories τ_i

for each action a_i, LGP optimizes over the full trajectory ξ, aiming towards optimal results. The results of the feasibility check of the two optimization layers are sent to the symbolic level to verify the action sequence generated π. The breakdown of keyframes and full path planning allows for a comprehensive optimization that maintains a balance between computational efficiency and the quality of the solution. By integrating task and motion planning in this way, LGP offers a powerful tool for robotics applications that require both high-level decision-making and precise control over movement, effectively advancing the state of the art in robotic planning.

2.2 Task Planning

2.2.1 Problem Description

Task planning in robotics involves generating a sequence of symbolic actions to achieve a goal within an environment. The problem involves a robotic agent assigned to perform task $\mathcal{T}$. Actions available to the robot are denoted by $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, each capable of altering the environment and the robot's state. The state of the system, S, includes both the robot's state and the environment's state, changing via the state transition function $\delta : S \times A \to S$ upon performing an action A.

The objective is to find a sequence of actions $(a_1, a_2, \ldots, a_n)$ that transitions the initial state s_0 to a goal state s_g where task $\mathcal{T}$ is satisfied. Determining the action chain requires the task planner to understand and proceed relevant effects of action on the environment such that they match the predicates of other actions. Other constraints include the robot's ability to perform only one action at a time, a cost associated with each action, and the necessity to avoid unsafe states or actions that could lead to failure.

2.2.2 Formal Planning Language

Robotic planning problems require formal languages to specify actions, states, and goals in a way that automated systems can reason about. First-Order Logic (FOL) [4, 5], or predicate logic, is a formal system that enables the representation of

objects, predicates, and quantifiers to express general truths about the world. In the context of planning, FOL provides the expressive power to define states and actions with preconditions and effects.

Building on FOL, the Stanford Research Institute Problem Solver (STRIPS) [6] language was introduced to model classical planning problems. STRIPS restricts FOL to a more computationally tractable subset by using a structured representation of the world in terms of predicates and finite sets of actions. Each action in STRIPS is defined by:

- An initial world models describing the current state of the world.
- A set of operators including description of their preconditions and effects.
- A goal condition.

While STRIPS simplified reasoning about actions and states, it lacked the expressiveness needed for more complex domains. To address this limitation, the Planning Domain Definition Language (PDDL) was introduced [7]. PDDL generalizes STRIPS by supporting advanced features such as conditional effects, universal quantification over dynamic universes (e.g., object creation and destruction), domain axioms over stratified theories, safety constraints, hierarchical actions composed of subactions and subgoals, and the management of multiple problems across multiple domains. PDDL separates the specification into two parts: a domain description and a problem description.

- **Domain description**: defines the reusable problem domain:
  - Requirements
  - Types and constants
  - Predicates
  - Action definition with parameters, preconditions, and effects
- **Problem description**: defines a specific planning instance:
  - The domain it belongs to
  - Object declarations
  - Initial conditions
  - Goal states

PDDL has become the standard for representing planning domains and problems, particularly in robotics and automated planning research. LGP [3] also implements a subset of PDDL to describe high-level symbolic planning components.

2.2.3 Tree Search

Tree search algorithms are a fundamental approach to task planning in robotics, leveraging systematic exploration of the action space to identify a sequence of actions that lead to the desired goal state. They operate by exploring the nodes of a tree, starting from the root and progressively visiting branches according to a specific strategy. The primary goal of a tree search is to construct a search tree where each node represents a state of the system S resulting from the application of a sequence of actions from the set $\mathcal{A}$.

In a typical tree search, each node N in the search tree can be defined as a tuple:

$$N = (S, \text{path}, \text{cost}),$$

where:

- S is the current state of the system,
- path is the sequence of actions taken to reach state S,
- cost represents the cumulative cost incurred by executing the actions in the path.

The root of the tree represents the initial state s_0, and each child node represents a new state generated by applying an action $A_i \in \mathcal{A}$ to its parent node's state.

Incorporating Heuristics in Tree Search

In heuristic search, a heuristic function $h(s)$ is used to estimate the cost of reaching the goal state s_g from the current state s. The heuristic should be admissible, meaning it never overestimates the true cost. This property ensures that the search algorithm remains optimal. Incorporating heuristics into the tree search process allows for more informed decisions about which paths to explore. For example, an informed search strategy like A* uses heuristics to prioritize nodes that are more

likely to lead to an optimal solution. Tree search algorithms with effective heuristic functions enhance the efficiency of task planning and provide a powerful mechanism for robotic agents to achieve complex tasks autonomously in dynamic environments.

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) [8] is an advanced tree search algorithm that has gained significant popularity for its effectiveness in complex decision-making environments, such as those found in robotic task planning. Unlike traditional tree search methods, MCTS uses simulations to generate more promising search trees, focusing on areas that have the potential to yield better outcomes.

MCTS consists of four primary phases in each iteration, which is visualized in Fig. 2.3:

- **Selection**: Starting from the root node, the algorithm progresses down the tree based on a selection policy to select the promising node.
- **Expansion**: The child node is added and we expand the tree considering available actions from the current state.
- **Simulation**: From the new nodes, the algorithm simulates outcomes of possible action sequences randomly until a predefined condition (like a maximum depth or a terminal state) is met.
- **Backpropagation**: The results of the simulation are propagated back up the tree, updating the statistical information at each node up to the root node.

In robotics, MCTS can be particularly useful for planning tasks that involve uncertainty or a large number of potential actions at each step. The algorithm's ability to focus on more promising parts of the search space without needing to exhaustively explore less relevant areas makes it well-suited for dynamic and complex environments. Heuristic-enhanced MCTS can prioritize exploring paths that appear safer or more efficient based on past experiences or sensor inputs. This method not only speeds up the planning process but also improves the reliability and safety of the robot's operations in real-world tasks.

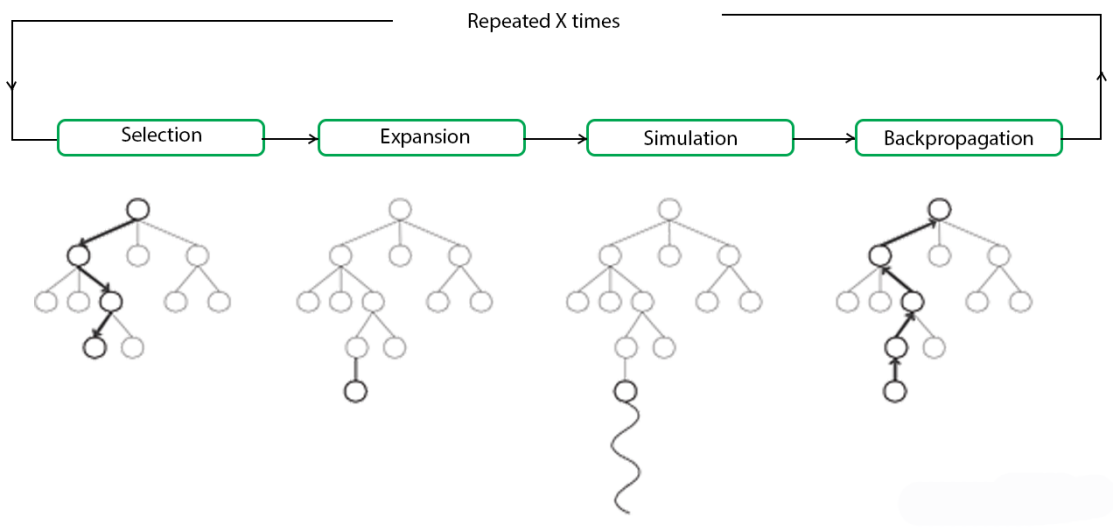


Figure 2.3: Diagram of MCTS process

2.2.4 Large Language Models (LLMs)

LLMs are a class of artificial intelligence models designed to understand and generate human-like text based on the training data they have been fed. LLMs are typically built on transformer architectures, which leverage self-attention mechanisms to process sequences of data (words, in the case of language). This architecture allows the model to weigh the importance of different words in a sentence, regardless of their position. This approach enables LLMs to capture nuanced meanings and generate coherent and contextually relevant text across a wide array of topics. These models are trained using vast amounts of text data through a process called unsupervised learning, where the model learns to predict parts of the text from other parts. The training involves adjusting millions (or billions) of parameters, which define how the model processes and generates language. As LLMs are trained on diverse internet text, they develop a broad understanding of natural language, increasingly integral to different applications, and pushing the boundaries of how machines understand and interact with human.

LLM has been integrated into TAMP as it can reduce the burden of defining TAMP problems in formal planning languages presented in Sec. 2.2.2. Given a

natural language description $\mathcal{L}$, the objective is to generate an action sequence $\pi = \{a_1, a_2, \ldots, a_n\}$ such that the symbolic task plan achieves the goal described in $\mathcal{L}$. It is essential for LLM models to understand the task in the context of robotics, and plan meaningful actions.

2.3 Motion Planning

2.3.1 Problem Description

Motion planning refers to the task of finding a sequence of configurations for a robot system to complete the action sequence, where configurations reflect the robot's degrees-of-freedom (DoF). In our mobile manipulator systems, the configuration space $\mathcal{C}$ includes the robot's base configuration for locomotion and the arm configuration for manipulation:

$$\mathcal{C} = \mathcal{C}_{\text{base}} \times \mathcal{C}_{\text{arm}},$$

where $\mathcal{C}_{\text{base}}$ represents the base's position and orientation in the environment, and $\mathcal{C}_{\text{arm}}$ represents the joint angles of the manipulator.

In robotics motion planning, several constraints and features are targeted to ensure safe, efficient, and feasible paths for a robot to navigate through its environment. The following list explains the contraints targetted during the development of this thesis:

- Geometric constraints: The robot must not collide with any obstacles in the environment. This requires checking for intersections between the robot's configuration and any obstacles at each point along the path. In addition, the robot must operate within certain physical boundaries or limits of the workspace, which ensures that the robot does not move outside the allowable region.
- Kinematic constraints: For robotic manipulators, each joint has physical limits on its range of motion, which must not be exceeded. These are often expressed as bounds on joint angles or positions.

- Dynamic constraints: Dynamic constraints take into account the robot's inertia, forces, and torques, ensuring that the robot's motions are physically feasible given its dynamics. Particularly, in legged manipulation, the stability of the robot system must also be considered.
- Task-specific constraints: Depending on the task, there might be additional constraints. For example, in manipulation tasks, there could be constraints on the end-effector's position, orientation, or force applied to objects. Path smoothness, energy and time efficiency are other possible task requirements.

2.3.2 Sampling-based Approach

Sampling-based algorithms find feasible paths by exploring samples instead of explicitly constructing the entire space. In this section, we provide preliminaries of one of the most traditional sampling-based motion planning algorithms, Probabilistic Roadmap, which is used in the thesis.

Probabilistic Roadmap

The Probabilistic Roadmap (PRM)'s process is visualized in Fig. 2.4, with two main phases:

1. **Construction phase**: A roadmap is built by sampling random points from the Cartesian space $\mathbf{x}_i$. The sampled configuration is then checked for feasibility, and added to the roadmap if it is valid. Next, valid configurations are connected by creating edges $(\mathbf{x}_i, \mathbf{x}_j)$ if there exists a collision-free path between them. Let $G = (V, E)$ represent the roadmap graph, where V is the set of valid configurations and E are the edges connecting them. The resulting roadmap serves as a connectivity structure that enables efficient path planning.

2. **Query phase**: Given a start and goal configuration, a search algorithm (e.g., A*, Dijkstra) is used to find a path through the roadmap. The algorithm attempts to find a feasible path τ from start ($\mathbf{x}_{\text{start}}$) to goal ($\mathbf{x}_{\text{goal}}$) such that:

$$\tau : [0, T] \to \mathcal{C}_{\text{free}}, \quad \tau(0) = \mathbf{x}_{\text{start}}, \quad \tau(T) = \mathbf{x}_{\text{goal}}.$$

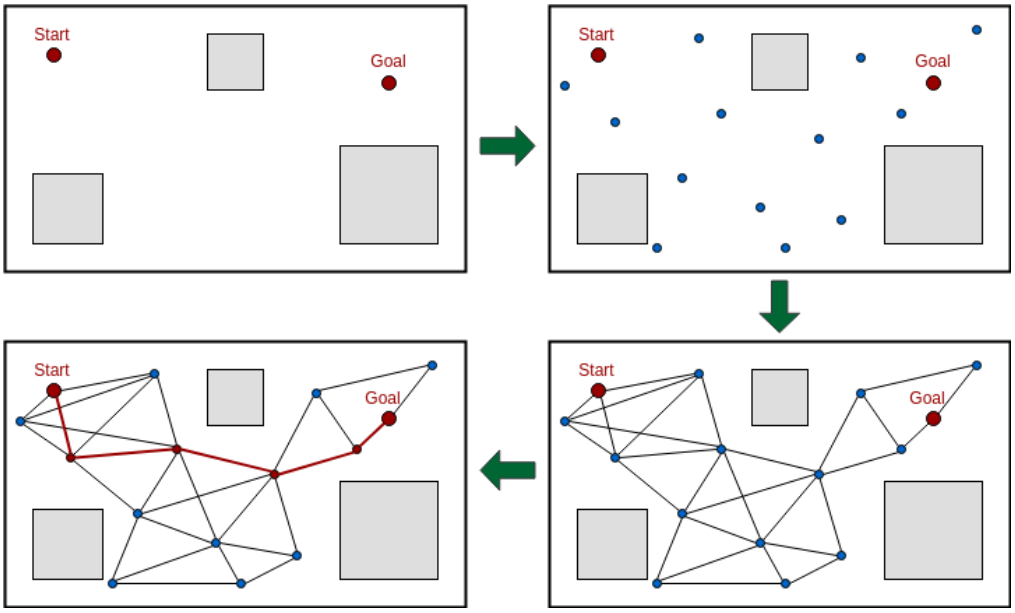


Figure 2.4: General process of sampling-based planners.

Sampling-based planners particularly excel in high-dimensional configuration spaces where traditional grid-based methods become impractical due to the extensive computations needed to decompose the entire space. These planners focus on a smaller set of randomly sampled configurations, making them feasible for scenarios where grid-based methods are not. Short planning time is another key feature of sampling-based algorithms, which bypasses the time-consuming space decomposition required by grid-based approaches.

However, these advantages come with trade-offs. Although sampling-based approaches can be extremely efficient due to not requiring a full exploration of the configuration space, this efficiency often comes at the expense of the solution's quality. The solutions from sampling-based planners are often less optimal because fewer samples mean reduced accuracy. There is a risk of not finding a path even if there is one exists. The randomness in sampling leads to variability in solutions for the same problem, though caching successful paths can mitigate this issue. Moreover, these planners struggle with navigating through narrow gaps due to the low probability of randomly sampling the precise configurations needed for such tight passages.

2.3.3 Optimization-based Approach

Optimization is the process of finding the best solution from a set of feasible solutions for a given problem, typically by maximizing or minimizing a particular objective function. This section describes both solving inverse kinematics (IK) and trajectory planning with optimization. In our projects, IK is used as a simplified method for checking feasibility, while trajectory optimization outputs the full path in configurations to perform TAMP solutions.

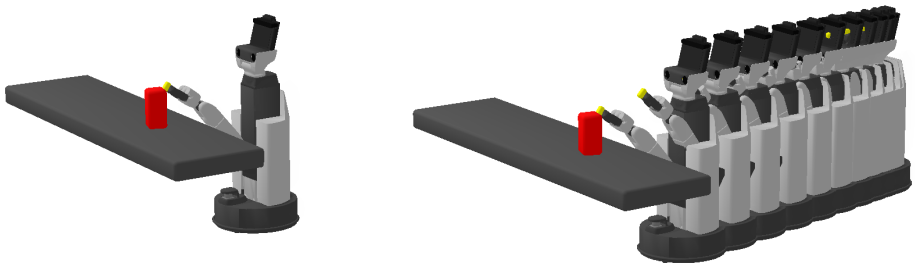
Inverse Kinematics (IK)

Inverse kinematics (IK)'s goal is to determine the necessary joint parameters that achieve a desired end-effector position and orientation (Fig. 2.5a). Unlike forward kinematics, where the pose of the end-effector is determined from given joint angles, IK involves finding joint configurations that meet specific positional criteria. Formulating IK as an optimization problem has become a prevalent approach due to its flexibility and effectiveness in handling complex kinematic chains with high-DoF systems. When there are multiple configuration solutions that leads to the same desired end effector position, optimization-based approach allow us to define an objective function, prioritizing custom cost and constraints (e.g. close to the nominal pose for stability).

The primary objective in IK is to find the joint angles or positions $\mathbf{q}$ that minimize the difference between the end-effector's actual position $\mathbf{x}(\mathbf{q})$ and its desired position $\mathbf{x}_d$. Mathematically, the IK problem can be formulated as:

$$\min_{\mathbf{q}} \|\mathbf{x}(\mathbf{q}) - \mathbf{x}_d\|^2$$

where $\mathbf{x}(\mathbf{q})$ represents the forward kinematics function, mapping joint variables $\mathbf{q}$ to the position and orientation of the end-effector, and $\|\cdot\|^2$ denotes the squared Euclidean norm.



(a) Inverse kinematics **(b)** Trajectory

Figure 2.5: Examples of motion plans in reaching an object.

Trajectory Optimization

Trajectory optimization involves the generation of optimal motion paths for robotic systems subject to constraints and objectives. An example of a whole-body trajectory planning solution of reaching towards an object is shown in Fig. 2.5b. Trajectory optimization formulates the motion planning problem as an optimization task where the goal is to find a sequence of trajectories $\{\tau_1, \tau_2, \ldots, \tau_n\}$ over a time interval $[0, T]$.

The complete trajectory optimization problem can be stated as follows:

$$\begin{aligned} \text{minimize} \quad & \int_0^T c(\mathbf{q}(t), \dot{\mathbf{q}}(t))\, dt \\ \text{subject to} \quad & \mathbf{h}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) = 0 \quad \forall t \in [0, T], \\ & \mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \leq 0 \quad \forall t \in [0, T]. \end{aligned}$$

Here, $\mathbf{q}(t)$ represents the robot's configuration at time t, and $\dot{\mathbf{q}}(t)$ is its time derivative. The equality constraints $\mathbf{h}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) = 0$ typically denote task-specific constraints, such as maintaining a desired position, velocity or force value. Meanwhile, the inequality constraints $\mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t) \leq 0$ ensure safety of the system, such as respecting joint limits and avoiding collisions with obstacles. Optimization methods iteratively refine the path to minimize the cost function while satisfying these constraints.

K-Order Markov Optimization (KOMO) [9], implemented as the motion planning method in LGP [3], is special among available trajectory optimization solutions in that it allows for optimizing over sequences where the structure and DoF of configurations changes over time. The method formulates trajectory optimization

as a k-order non-linear sum-of-squares constrained problem over a time discretized trajectory, where the k-order cost vectors c_t flexibly enables representation for both transition and task-related costs:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=0}^{T} c_t(\mathbf{q}_{t-k:t})^\top c_t(\mathbf{q}_{t-k:t}) \\ \text{subject to} \quad & \mathbf{h}_t(\mathbf{q}_{t-k:t}) = 0 \quad \forall t \in [0, T], \\ & \mathbf{g}_t(\mathbf{q}_{t-k:t}) \leq 0 \quad \forall t \in [0, T]. \end{aligned}$$

2.4 Perception in TAMP

In robotic applications, perception plays a crucial role in providing robustness and reactivity to the robot's planning and execution. Although perception is not the focus of this thesis, visual data is processed to extract relevant information such as object detection, localization, and scene understanding, enabling robots to make informed decisions and perform tasks within the environment. Particularly, in our context of task and motion planning for mobile manipulators, perception modules involve extracting information from raw RGB-D cameras and processing inputs for the main TAMP planner. This section delves into the fundamental components of our approach to TAMP perception that help a robot perform tasks autonomously.

2.4.1 Scene Processing

This section describes our surface perception methods for the project in Chapter 4, which is essential to achieve autonomy in remote areas. It enables our specialized planning framework to operate in a task-aware way.

Scene Reconstruction

Scene reconstruction focuses on creating a 3D model of the operating environment. A widely used approach is the Truncated Signed Distance Function (TSDF) [10], which provides a robust framework for integrating multiple depth images into a single volumetric model. The TSDF representation maps each point in space to its shortest distance to a surface boundary, distinguishing between points inside and

outside objects with positive and negative values, respectively. The mathematical formulation of TSDF for a point $\mathbf{p}$ in space is given by:

$$TSDF(\mathbf{p}) = \min\left(\frac{\text{distance}(\mathbf{p}, \text{surface})}{\text{threshold}}, 1\right)$$

This function is truncated at a threshold distance to maintain computational efficiency and to handle noise and outliers in depth measurements.

Segmentation

Segmentation partitions the reconstructed scene into meaningful clusters, typically representing individual objects or parts of objects. Techniques such as Euclidean clustering [11] and Random Sample Consensus (RANSAC) [12] are employed in our project:

- **Euclidean clustering**: a form of spatial clustering, which involves grouping points that are spatially close, assuming that points that are near each other belong to the same object. This technique groups points based on their Euclidean distance, assuming points that are spatially close belong to the same object. The process begins by defining a distance threshold, which is the maximum distance between points for them to be considered part of the same cluster. Using a k-d tree for efficient search, the algorithm forms clusters by starting with a seed point and recursively adding neighboring points within the threshold distance until no more points can be added, then moves on to the next unclustered seed point.

  Euclidean clustering is advantageous as it does not require prior knowledge of the number of clusters and can adapt to the shape of objects, making it ideal for segmenting complex and irregular structures in robotic tasks. However, setting the correct distance threshold is critical, as too small a threshold can lead to over-segmentation, while too large may merge separate objects into a single cluster, necessitating careful tuning based on the scale and density of the point cloud to optimize segmentation outcomes.

- **RANSAC**: Useful for identifying and isolating geometric shapes (e.g., planes, cylinders, spheres) within noisy data. RANSAC operates by randomly selecting a subset of the data to hypothesize a model, then determining how many other data points fit this model within a predefined tolerance, classifying them as inliers. The process is formulated as:

$$\text{Model} = \arg\max_{m} \sum_{i=1}^{N} \mathbf{1}(\text{distance}(\mathbf{p}_i, m) < t),$$

where $\mathbf{1}$ is the indicator function, t is a threshold, $\mathbf{p}_i$ are data points, and m is the estimated model parameters.

The key advantage of RANSAC is its robustness against outliers, making it particularly suitable for environments where the sensor data may contain a significant amount of noise. The method's effectiveness, however, depends heavily on the appropriate setting of tolerance thresholds and the number of iterations, which must be balanced to achieve reliable detection without excessive computational cost.

2.4.2 Object Detection

Object detection is a critical component of robotic perception, enabling the identification and localization of objects within an environment. This process is fundamental for interaction and manipulation tasks performed by mobile manipulators with general tasks. Our pipeline integrates deep learning methods, particularly YOLO [13], which is a popular method in real-time object detection. YOLO frames object detection as a single regression problem, eliminating the need for sliding windows or region proposal methods. To analyze the entire image at once, YOLO divides it into a grid and predicts both bounding boxes and class probabilities for each section simultaneously. This innovative method enables YOLO to operate with high speed and accuracy, making it ideal for real-time applications like autonomous vehicles, video monitoring, and robotics.

2.5 Hardware Platforms: Robots And Sensors

In this section, we review the hardware platforms used for the development of this thesis. The research development was done on either legged or wheeled manipulators, depending on the specific application and environmental requirements developed in each project. Additionally, we also mentioned the cameras used on each robot platform to obtain environmental perception.

2.5.1 Legged Mobile Manipulator

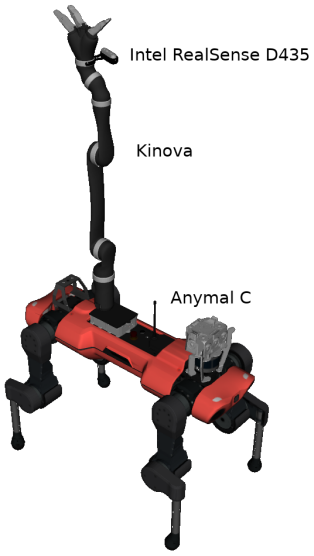
A legged manipulator combines the mobility of a legged robot with the dexterity of a robotic arm, creating a versatile platform capable of both locomotion and manipulation tasks. Similar to legged robots, it utilizes limbs for movement, enabling navigation through diverse and complex environments such as rough terrain, stairs, and obstacles. At the same time, the manipulator with its articulated joints and end-effector can easily grasp, manipulate, and interact with objects in its surroundings. This dual capability makes legged manipulators highly adaptable for applications where both mobility and manipulation are crucial, such as disaster response, exploration in challenging terrains, and tasks requiring object handling in unstructured environments.

In our work, we use a lightweight 6-DoF arm with a quadrupedal base. This combination provides a stable and flexible legged manipulator in real-world settings. Fig. 2.6 presents the legged mobile manipulator used in this work, where we put a Kinova arm on an ANYmal C. The main specifications of the robots are shown in the first two rows in Table 2.1, which indicate that the lightweight Kinova arm can be mounted on the Anymal C, respecting the base's payload. On our legged platform, we attached an RGBD Intel Realsense camera on the wrist of the Kinova arm. This camera is equipped with depth sensors based on active stereo technology, allowing them to capture 3D information with high accuracy and precision.

(a) Kinova

(b) ANYmal C

(c) RGBD Intel Realsense camera



(d) Kinova on ANYmal C

Figure 2.6: Our legged manipulator platform includes the Kinova arm (a) on the ANYmal C quadrupedal base (b), with a RGBD Intel Realsense camera (c) mounted on the wrist of the arm.

2.5.2 Wheeled Mobile Manipulator

A wheeled manipulator combines the agility and speed of wheeled robots with the precision and versatility of a robotic arm. This robotic system is equipped with wheels for swift movement across flat and smooth surfaces, making it highly efficient in environments such as warehouses, factories, and homes, where rapid navigation is essential. At the same time, it features articulated joints and grippers akin to traditional robotic arms, enabling it to handle objects, perform manipulation tasks, and interact with its surroundings with accuracy and control.

Name	Type	Size (LxWxH)	Weight	Payload	Max speed	Reference
Kinova	Robotic arm	900 mm*	5.2 kg	1.3 kg	20 cm/s	Link
ANYmal C	Legged robot	1054 x 520 x 830 mm	50kg	10 kg	1.0 m/s	Link
HSR	Wheeled manipulator	430 x 430 x 1005 mm	37 kg	1.2 kg	0.8 km/h	Link
PR2	Wheeled manipulator	668 x 668x 1650 mm	226.8kg	1.8kg	3.6km/h	Link

* Maximum reach

Table 2.1: Specifications of the robotic platforms in this thesis.

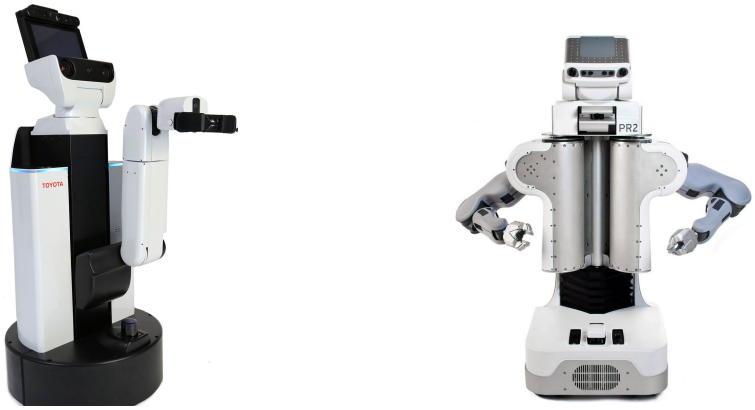
The combination of mobility and manipulation enhances operational productivity while maintaining safety, offering a versatile solution for a wide range of industrial and service applications.

Fig. 2.7 shows the wheel manipulators used to develop our domestic TAMP planner, which include the Toyota Human Support Robot (HSR) [14] and the Personal Robot 2 (PR2) [15]. The robots' specifications are shown on the last two rows of Table 2.1.

- HSR is equipped with a 3-DoF base and 5-DoF arm, enabling dexterous behaviours. The end effector of the HSR is a two-fingered force-compliant gripper with an RGB camera. The head has 2 DoFs, tilt and pan. There are three camera sensors: RGB-D Asus Xtion Pro Live, a stereo camera and a wide-angle camera.

- PR2 is equipped with an omnidirectional base that provides 3 DoFs for smooth navigation, and two 7-DoF arms for dexterous manipulation tasks. Each arm is fitted with a parallel gripper capable of grasping a variety of objects. The head has 2 DoFs, with pan and tilt capabilities. The PR2 is equipped with multiple sensors, including an RGB-D camera (Kinect) mounted on the head,

a laser rangefinder, and stereo cameras. As our lab does not have access to a physical PR2 robot, the pick-and-place and sorting experiments with the PR2 in Chapter 5 are conducted in simulation.



(a) HSR **(b)** PR2

Figure 2.7: Wheeled manipulators platform

3
Literature Review

Contents

This chapter provides a literature review of existing methods in TAMP and its individual components. Section 3.1 presents how classical planning has been studied in the context of robotics planning. Section 3.2 reviews current motion planning approaches in mobile manipulation. Section 3.3 focuses on the state-of-the-art (SOTA) integration of task and motion planning in robotics.

3.1 Classical Task Planning in Robotics

Classical planning techniques [16] have been the foundation of influential planning approaches in robotics, including state-space search, heuristic-based planning, and symbolic reasoning, which have shown notable applicability in robotics applications. One of the earliest approaches to task planning, state-space search explores the

space of all possible actions to determine an optimal or feasible path to a goal state. In robotics, this approach requires searching through the state space in controlled environments to identify a sequence of actions that the robot can execute to achieve its goals, such as navigating to a target position or performing a series of manipulation tasks. The foundational concepts of robotic planning were rooted in state-space search algorithms, including breadth-first search (BFS) [17], depth-first search (DFS) [18], and best-first search [19]. BFS and DFS, developed in the 1950s, introduced structured ways to explore possible states by either expanding outwards in levels (BFS) or diving deeply along one path before backtracking (DFS). Despite being general AI techniques initially, these methods were soon applied in robotics, where robots needed systematic strategies to explore and navigate environments. For instance, BFS was particularly effective in grid-based environments, enabling robots to avoid obstacles and find paths to goals in structured settings [20]. DFS was also studied in maze traversal problem with a humanoid robot [21]. However, these early techniques were limited by high memory requirements and inefficiencies in complex, high-dimensional spaces. The introduction of best-first search, specifically A* [22], marked a significant advancement in classical planning. Unlike the popular Dijkstra [23] path-finding algorithm, which greedily explores the weighted graph for the lowest cost, A* uses heuristics to prioritize paths closer to the goal, optimizing search efficiency. In robotics, A* has been instrumental for pathfinding in grid-based maps, where it enables robots to avoid obstacles and find optimal paths to targets [24–26]. However, the limitations of the above approaches became apparent in dynamic, high-dimensional environments, where they struggled to handle continuously changing variables. D* (Dynamic A*) has been adapted to address path planning in dynamic environments where obstacles may change, enabling robots to adjust their paths in real time [27].

In 1971, the STanford Research Institute Problem Solver (STRIPS) [6] represented a milestone in robotics planning. Unlike previous search-based methods that primarily addressed pathfinding, STRIPS introduced a more structured approach to planning tasks by defining actions, states, and goals in terms of predicates,

allowing planners to handle more abstract tasks. This work introduces the concept of action-centric representation and lays the foundation for Planning Domain Definition Language (PDDL) [7], a well-known and widely used standard in planning model definition. PDDL is an expressive language that is not case-sensitive and can compactly capture physics behaviours, therefore helping to bridge the gap between research and applications. PDDL also supported multi-agent planning, which extended robotic task planning beyond single-agent tasks to scenarios requiring collaboration. With the emergence of artificial intelligence planning research, there are now many variants of PDDL introduced serving different focuses [28–31]. One popular version of PDDL is 2.1 [29], which develops temporal and metric features with the ability to model non-binary resources and continuous plans. Apart from FOL used in STRIPS and PDDL, there are higher-order logic formulations that include predicates of predicates, however, these methods are relatively computationally expensive to be used in TAMP. Answer Set Programming (ASP) [32] also describes the planning model with logic programs, offering a declarative approach that contrasts with the predicate and function-based structures of FOL and the action-centric syntax of PDDL. Jiang et al. [33] compared ASP with PDDL and pointed out that PDDL-based planners performed well in solving multi-step problems, while ASP-based methods are suitable for dealing with a large domain or when each step in the plan demands considerable reasoning to infer the world state.

Hierarchical Task Network (HTN) [34] solved planning problems by decomposing high-level tasks into sub-tasks. The hierarchical structure allowed robots to manage complex goals by breaking them down into smaller, manageable components. HTN planning was widely adopted in robotic applications where task complexity required structured organization, enabling robots to follow intricate workflows [35, 36]. HTN can also scale up to solve multi-agent problems [37, 38] or human-robot interaction [39], and was deployed as the task planner in TAMP [40]. In addition to using HTN as a sole task planner, researchers also consider HTN as a hierarchical planning solution for TAMP [41].

In recent years, the integration of machine learning with classical planning has led to the development of robust, adaptive planning systems. Learning-based methods, such as reinforcement learning (RL), significantly enhance the flexibility of planning algorithms by enabling robots to learn and adapt optimal sequences of actions in response to environmental changes. These hybrid approaches retain the structure and efficiency of classical planners while introducing adaptability crucial for real-world applications. For instance, RL has been combined with A* to help robots plan navigation paths in partially known environments, adapting to uneven terrains [42]. On the other hand, in [36], the authors integrate online learning into an A*-based HTN planner to find the plan of maximal expected utility, showcasing the potential of these integrations to enhance robotic autonomy and decision-making.

3.2 Motion Planning in Mobile Manipulation

Mobile manipulation combines movement and object interaction, allowing robots to perform complex tasks that require both mobility and precise manipulation, such as picking, placing, and assembling. Whether mounted on wheeled or legged bases, mobile manipulators face unique motion planning challenges in navigating their environments and ensuring effective reach and manipulation of objects [43]. Effective motion planning for mobile manipulators thus involves not only pathfinding but also selecting optimal base placements for efficient and stable task execution. Often, locomotion and manipulation are treated separately in controlling these models due to their complicated dynamics. For example, the authors in [44] combine A* with RRT methods for the mobile base and manipulator motion planning respectively. The framework processes manipulation actions conditioned on the planned navigation solution. Iriondo et al. [45] instead proposed a learning-based approach that generates the mobile base solution with deep RL with the arm planner's feedback. However, the system requires tuning for different scenarios and the learnt policies are not reproducible. Inverse reachability map (IRM) is notable for its effectiveness in capturing feasible base locations from which a robot can achieve specific end-effector positions. An IRM represents a precomputed map

that allows the robot to identify optimal base positions that maximize reachability for manipulation tasks [46]. While collision is initially excluded in this method [47, 48], Merkt et al. [49] used inverse Dynamic Reachability Maps [50] for optimal base placement and continuous scene monitoring and replanning to robustly accommodate dynamic changes.

Optimization is an effective tool that has been developed to calculate either base placement [51, 52] or whole-body control signals [53, 54] in loco-manipulation systems. Whole-body motion planning is an optimal way to make use of the manipulation capabilities of these high-DoF robots. The Stability and Task Oriented Receding-Horizon Motion and Manipulation Autonomous Planner (STORMMAP) [54] uses a quintic spline to parameterize an optimization problem to achieve real-time whole-body force plans on a quadrupedal manipulator. While contact forces with the environment during manipulation tasks can cause instability in legged torsos, Ferrolho et al. [55] handled such external disturbances using a trajectory optimization method and optimized whole-body loco-manipulation plans for robustness from unknown and known disturbance directions. This motion planner considers the full dynamics of the platform, therefore, can effectively plan when robot feet break contact with the environment. The authors in [53] formulated a hierarchical optimization algorithm as a whole-body planning and control framework to generate torque control signals. This work focuses on robustness of whole-body control and takes into account dynamics during manipulation tasks. In [56], the authors introduced a whole-body optimization framework on a wheeled manipulator that targets task-specific constraints. The algorithm generates motion plans from the integrated inputs including state estimation, perception and users. Instead of optimization, Abdessemed [57] developed the forward kinematics equation of a mobile manipulator in the form of a pseudo-neural network, which was then used to determine the configurations for a specified end-effector trajectory. The method back-propagates EE errors to update the weight for next configuration. Sampling-based approaches were also introduced in high DOF motion planning [58, 59]. Yang et al. [58] modified RRT-Connect planner [60] to handle time-indexed constraints.

The approach plans in configuration space and can produce collision-free motion planning in dynamic environments. However, these sampling-based techniques do not guarantee optimality of smoothness and can cause jerky motions.

3.3 Task and Motion Planning

Research in TAMP primarily seeks solutions on how to interleave causal and geometric requirements in order to complete the goals. The main challenge in combining task and motion planning is their hybrid nature. While task planning involves discrete task specifications, motion planning is solved in continuous space.

3.3.1 Problem Specification in TAMP

Effective TAMP systems require not only planning algorithms but also a representation framework that supports the interaction between high-level task logic and continuous motion constraints. To simplify joint reasoning across task and motion planning, semantic attachments [61, 62] implement external black-box procedural reasoning to check for kinematics, dynamics or geometric constraints. Accordingly, instead of looking up state transitions, the method provides the integration of external functions for action specification. This approach is restricted to domains with pre-specified discrete set of action poses (e.g. grasp poses). Meanwhile, PDDLStream [31] introduces *streams* to dynamically generate necessary geometric data during planning, which includes procedural and declarative elements. The procedural component is a conditional generator, a function that maps input values to a potentially infinite sequence of output values. These generators are used to produce new values based on existing ones, such as generating robot configurations that meet certain kinematic constraints given a specific pose or grasp. At the same time, the declarative element defines the logical facts that its inputs and outputs must satisfy. This dual structure allows planners to reason symbolically about the relationships between inputs and outputs, while abstracting away the internal mechanics of how those outputs are computed. In this way, PDDLStream defines a problem interface where symbolic operators are linked to external samplers

and validators. This approach builds upon earlier work such as STRIPStream [63], which introduced similar stream abstractions in the context of STRIPS-based planning. The formalisms enable compact problem specification in domains with high-dimensional continuous variables and geometric evaluations. Other approaches attempt to model TAMP to handle directly in continuous space [3, 64, 65]. Notably, LGP [3] specifies TAMP problem as a search over symbolic skeletons, then conditions the full trajectory to solve a nonlinear constrained optimization problem in continuous space. By specifying TAMP problems through a tightly integrated structure, LGP targets optimal TAMP solution.

3.3.2 Satisfactory TAMP

Satisfactory TAMP focuses on generating feasible solutions, ensuring task completion without necessarily optimizing path or task efficiency. Task and motion planners are traditionally combined on a level basis, which can be decoupled or integrated. Accordingly, each layer will keep its own level of abstraction and the TAMP model should be able to translate knowledge between the two levels. Given the introduction of STRIPS [6], robot planning was initially studied with a decoupled hierarchical structure [66]. In this work, Shakey the robot plans a high-level action, then sends it to the motion control layer for execution and has no alternative solutions when the movement is infeasible, as denoted in Diagram 1 of Figure 3.1. This approach is not really practical due to the strong assumption that every discrete plan can be refined into a correct robot configuration. The hierarchical structure was also utilized in [67] to decompose task dependencies, and the symbolic search initially assumed all valid paths.

Due to robot's high dimensionality and dynamics, more integrated TAMP [68] considers failures and has the ability to replan to make sure that the strategy is executable. Common methods are taking one of the two parts as the base plan and solving the other level accordingly. For example, Srivastava et al. [69] introduce a TAMP method that backtracks and finds alternative task plans if the motion solver fails, as visualized in Diagram 2a in Figure 3.1. Such approach usually discretizes

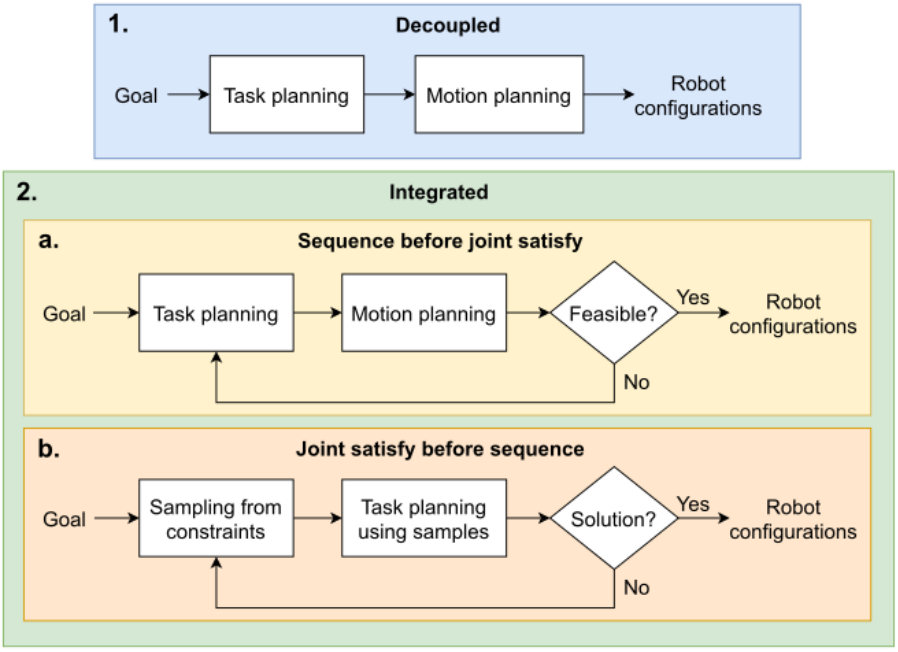


Figure 3.1: Satisfactory TAMP approaches.

the continuous space to bring it to symbolic planning. The method can utilize off-the-shelf planners and make use of SOTA techniques. TMKit [70] is introduced as the first open-source framework to implement such a structure, based on [71, 72]. Diagram 3b in Figure 3.1 shows an opposite approach, where the task search space is bounded by sampling the continuous workspace [73]. Given that task planning is usually in a finite domain, having to re-plan the task sequences is considered to produce a lower cost compared to continuously sampling the geometry constraints.

3.3.3 Optimal TAMP

Due to the complex nature, TAMP approaches usually tackle this problem separately and focus more on completeness. Solving such models optimally is difficult because the optima of each layer might not be the overall optimal solution. Kongming [74] was the first approach to integrate discrete and continuous action planning from an optimal perspective, where Unmanned Air Vehicles's mission is to perform a survey in algal bloom regions. The method combines a planning graph for discrete actions and flow tubes for continuous actions, which was modelled as a mixed logic linear (non-linear) program. The method is tied to a fixed time discretization, which is

difficult to apply on long-horizon tasks. However, it brings up the need for a tightly coupled task and motion planning framework in robotics applications.

In 2015, Toussaint [3] introduced LGP algorithm, which formulates TAMP as a mathematical program that uses optimization methods to find locally optimal plans instead of solely feasible ones. The task in this work is to build the highest stable tower from a list of blocks and boards. LGP generally defines three levels of approximation and switches between symbolic search and configuration optimization that is conditioned by symbolic decisions. While level 1 optimizes the final configuration, levels 2 and 3 optimize keyframes (e.g. grasp poses) and full path respectively. This paper implements FOL to formulate the planning problem and uses a relatively simple tree search method. By bringing logic into geometry, LGP does not need to arbitrarily discretize the continuous space and directly optimize continuous solutions. Variants of LGP papers include a heuristics method for long-horizon tasks [75, 76], a dynamic LGP for human motion prediction [77] or an approximation solver for cooperative manipulation [78]. Other optimization-based mathematical approaches to TAMP is constraint programming (CP) [79, 80] and mixed integer programming (MIP) [80–82]. Booth et al. [80] did a comparison between MIP and CP and concluded that inference-based CP is better than relaxation-based MIP in terms of time and solution quality. This work is tested on simulation and claims to be case-sensitive. MIP is originally a decision making and scheduling method that solves non-convex problems. When it comes to robotics planning, this optimization solution can help to include discrete decisions with integer variables. Two common approaches to solving MIP are branch-and-bound [83] and cutting plane [84]. Deits et al. [85] introduced a planner that uses MIP to generate globally optimal sequences of footsteps in difficult terrain. Multi-robot task planning is a common application of MIP [86, 87]. In short, these mathematical techniques allow encoding logical decisions and geometric constraints in nonlinear optimization models without backtracking, targeting globally optimal strategies.

As more interest has been driven to solving TAMP optimally, besides mathematical programming, a very recent work from Thomason et al. [88] proposes

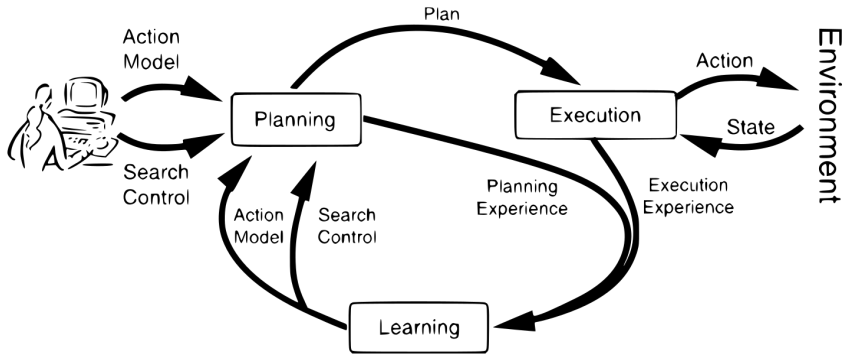


Figure 3.2: Learning integration in automated planning [89]

asymptotically optimal decision-making using informed tree search. The model effectively combines constraint-based symbolic planning, distance-based predicate representation, and batch-sampling-based optimal motion planning to solve a hybrid state space problem. Optimization-oriented TAMP often integrates cost-based methods for symbolic or motion planners to obtain local optimality, where task sequences are both feasible and cost-effective [31].

3.3.4 Machine Learning in TAMP

Classical Learning-based Approaches

Recent advancements in learning algorithms have significantly enhanced robotic capabilities, benefiting research in TAMP. In general, TAMP approaches that treat symbolic planning as a separate level can utilize SOTA machine learning techniques in classical planning [89]. Figure 3.2 shows a typical procedure for integrating learning in automated planning. The author in [89] indicates two main usages of machine learning, which are learning action models and/or learning search control. While action models are for obtaining the state transition of the world, search control helps to guide the planner's search for solutions.

Heuristics are a powerful approach to add to existing methods to get fast and robust strategies. Heuristic has been widely developed as an efficient tool in solving symbolic tree search, which is also known as informed search. Heuristics functions provide the search with the cost it takes to the final goal, with the purpose to guide the search. Learning-based heuristics can be well investigated to replace

hand-crafted ones in the planning domain. A popular example is AlphaGo [90], which is breakthrough research in using deep learning in classical planning. The method successfully combines supervised and reinforcement learning in Monte Carlo tree search to produce large state space decision-making. Inspired by this, Kim et al. [91] presented ADMON, an actor-critic algorithm to learn a policy from past experience and use it to guide the planner. The method focus on geometric-TAMP (G-TAMP), which decides object poses to account for high-level goals. Balac et al. [92] proposed a regression tree method to predict the effect of terrain on robot actions. The action cost from the low-level learning model is used as a parameter in the deliberative process in TAMP. In optimization-based LGP [77], motion prediction was also explored using recurrent networks along with reinforcement learning (RL) to accommodate human-robot coordination by utilizing the online MoGaze dataset [93]. This Dynamic LGP replans periodically and can produce long-term predictions.

The machine learning optimizer (MLOPT) [94, 95] was introduced to allow online solution to MIP problems with high speed, which outperformed the SOTA Gurobi solver [96]. Based on this work, Cauligi et al. [97] introduced CoCo (Combinatorial Offline, Convex Online) framework to enable real-time MIP TAMP solutions. The approach uses offline data to train a task-specific strategies classifier for plan prediction. However, the control system considered in the article is convex. Recent research from Funk et al. [98] defined a hierarchical structure for robot assembly discovery applications. The model uses block selection and final poses resulting from the high-level MIP-based formulation to guide the exploration of the reinforcement learning policy. The output sequences of action are then sent to the lowest level for motion planning. From a different perspective, Gupta et al. [99] developed a Graph Neural Networks (GNN) algorithm to replace the branching heuristic in the traditional branch-and-bound solver of MIP. Another machine learning approach to improving MIP solvers was proposed in [100] introducing variable branching, which can benefit TAMP problems with optimization-based MIP formulation.

AI methods have also been powerful in handling uncertainties from robot operations. Online learning is another approach to allow robust solutions from

TAMP. The technique has fewer assumptions in place and allows to accommodate contingencies during robot execution. An example can be found in [101], where the system assesses human intervention to regulate robot plans in human-robot collaborative tasks application. Apart from online learning, there are other learning-based approaches to tackle highly dynamic, partially observable, or non-deterministic working environments with Partially Observable Markov Decision Process (POMDP) [102, 103], including but not limited to robotics planning [104].

Foundation Models

Foundation models have recently emerged as a transformative force in TAMP, leveraging pre-trained, large-scale models that can be fine-tuned for specific robotic tasks. This approach shifts TAMP from task-specific training to broader generalization, where a single model can adapt to various planning scenarios. The application of LLM models in robotics, driven by recent advancements in natural language processing, has become a focal point of intensive research [105]. Silver et al. [106] integrates an LLM to interface with PDDL, showing that it can solve nontrivial tasks. The paper also concludes that LLM can be useful in guiding a heuristic search. Plansformer [107] requires PDDL domain inputs and fine-tunes the transformer model on planning problems. Although the PDDL interface is a neat approach to use with the current task planner, it is difficult for end-users to add new tasks. On the other hand, Liang et al. [108] utilize LLM for generating code policies that can be executed on the robot. Cascades of LLMs are also investigated to break down the levels of abstraction in robotics planning [109, 110]. RoboTool [109] integrates four LLM components in the planer: *Analyzer* identifies key concepts critical to task feasibility; *Planner* devises plans for tool use; *Calculator* determines parameters for the skills, and the *Coder* generates executable code. The pipeline targets TAMP problems that require tool usage, which proves to successfully select tools and sequence actions.

The ability to account for the robot's kinematics or geometry in robotics planning process is critical in real-world applications. SayCan [111] and Text2Motion [112]

are notable for considering geometric feasibility when planning the action sequence. SayCan [111] incorporates skills probabilities in LLM models in the form of value functions, allowing the models to have the grounding capability of robotics systems. The algorithm was tested intensively with complex and varied human language instructions, validating its high success rate in domestic robot planning. On the other hand, Inner Monologue [113] designs closed-loop language feedback from the environment in order to improve the robustness in completing the instructed tasks, allowing robots to handle disturbances. Vision-language models (VLM) [114–119] have also been extensively studied to explore how foundation models can be extended to cover both perception and task planning, with an extension to end-to-end vision-language-action research [120, 121]. For example, Contrastive Language-Image Pre-training (CLIP) [118] has been a popular and widely-used VLM model in robotics due to its powerful image reasoning capability. PIGINet [122] implements CLIP for generating text and image embeddings and introduces a transformer-based plan feasibility predictor to be integrated into a TAMP planner. The method fuses 6 cameras, making it harder to deploy in domestic settings. In 2023, PaLM-E [123] develops an embodied version of PaLM [124] by incorporating sensor data in a multi-modal structure, which generalizes well to embodied reasoning, vision-language and language tasks. Recently, Gao et al. [125] interestingly implemented LLM to improve physical understanding from vision, which successfully enhances success rate in object-centric planning, e.g. assessing material or fragility from visual appearance. However, the method doesn't cover actual physical quantities or feedback of the robot systems.

Research indicates that LLMs may not fully replicate human reasoning capabilities [126]. Consequently, integrating human-in-the-loop (HITL) that incorporates human intervention or oversight at key stages of automated processes proves essential for maintaining the reliability and safety of robotic systems [127]. Vemprala et al. [128] implement ChatGPT with HITL, emphasizing the dual benefits of enhanced model training through human feedback and increased safety during operations. Ren et al. [129] instead focus on human-robot interaction, where the robots

Figure 3.3: While LLMs are not grounded in the world environment, SayCan [111] has been a popular LLM-based robotics planner that considers geometric feasibility during planning with LLM.

decide whether or not to ask for human intervention. Human is also a source of feedback to Inner Monologue [113] to deal with ambiguous queries, enhancing human-robot interaction.

Often, LLM-based models are data-intensive, demanding substantial training datasets to perform optimally. In robotics, this demand poses a significant challenge, as collecting large, diverse, and representative datasets from physical robots is time-consuming, labor-intensive, and costly. For instance, RT series [120, 130] introduces an end-to-end approach to applying transformers to robotics, requiring a large amount of data collected in seventeen months to map language to actions. Similarly, Lynch et al. [131] proposes an interactive method for language-conditioned robot skills, compiling a dataset through 2.7k hours of collection on real robots. Prompt engineering, on the other hand, has shown promise in applying LLM to robotic tasks in a zero-shot manner [113, 132, 133]. However, it is not robust in real-world execution without properly understanding the operating environment. Fine-tuning methods involve adjusting the model's parameters with a custom dataset, which has been successful in capturing the large-scale pre-trained knowledge while ensuring it is adapted to certain requirements in task domains or formats [125, 134]. On the other hand, these large models, which enable embodied intelligence, rely heavily on cloud processing, making fully autonomous onboard operation impractical. Driess et al. [123] also reported that as model size increases, PaLM-E exhibits a marked reduction in forgetting of language capabilities from PaLM. These highlights a

trade-off between the scalability of language models and their deployment feasibility in resource-constrained environments.

3.4 Conclusion

In this chapter, we explored pipelines and methods of TAMP research. Developing TAMP involves the integration of various subsystems, ranging from understanding the environment to ensuring the execution of the generated solutions. Accordingly, our review covered the SOTA advancements in individual modules and fully integrated systems within this context.

The following three chapters will delve into specific contributions, introducing our TAMP solutions on legged and wheel manipulators, with both industrial and domestic environments where suitable. Each chapter is self-contained and includes a project-specific review of the relevant literature up to the time of the work's development. After presenting each contribution, we will reflect on its significance, and outline challenges that we address in later chapters or discussed in Chapter 7.

4

Hierarchical Optimization-based TAMP

In this first project, we look into how to solve specialized task with an optimization perspective. The project is developed with the following characteristics.

Task specification	Mathematical constraints
Interaction between task and motion planning layers	Decoupled
Optimality vs feasibility	Asymptotically optimal
Open-loop vs closed-loop	Open-loop
Role of human	Shared-autonomy

In particular, this project address a dexterous task in robotics planning with mobile manipulators. Motivated by the need for autonomous robot systems to perform offshore asset inspection, we develop an optimization-based planner as an end-to-end solution for coverage planning problems on a legged manipulator.

As the robot is expected to perform a coverage task in remote sites, we address autonomy in unknown environments. The developed framework is a perceptive planner with the enhanced capability of planning with complex and disjoint surfaces while guaranteeing reachability and stability of the robot system.

This chapter presents a novel two-layer hierarchical optimization structure comprising mission planning and motion planning, solving multi-surface coverage

planning for loco-manipulation systems.

- The framework effectively integrates perception, surface sequencing, coverage planning and motion planning to obtain end-to-end solutions.
- The multi-surface sequence is optimized using a mixed-integer linear programming (MILP) approach, while the motion planning phase solves a whole-body optimal control problem.
- The system also includes a user-friendly interface for efficient teleoperation, which is necessary in remote inspection.
- We showcase the versatility and scalability of our approach across various robotic coverage path planning tasks, including multi-surface asset inspection using a quadrupedal manipulator.

4.1 Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring

The article in this section was presented at the *IEEE International Conference on Automation Science and Engineering (CASE)* 2023 [135]. Our project website is available online at https://sites.google.com/view/multisurface-coverage-planning.

Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring

Kim Tien Ly, Matthew Munks, Wolfgang Merkt, Ioannis Havoutis

***Abstract*— Regular inspection and monitoring of aging assets are crucial to safe operation in industrial facilities, with remote robotic monitoring being a particularly promising approach for asset inspection. However, vessels, pipework, and surfaces to be monitored can follow complex 3D surfaces, and frequently no 3D as-built models exist. In this paper, we present an end-to-end solution that uses an optimization method for coverage path planning of multiple complex surfaces for mobile robot manipulators. The system includes a two-layer hierarchical structure of optimization: mission planning and motion planning. The surface sequence is optimized with a mixed-integer linear programming formulation while motion planning solves a whole-body optimal control problem considering the robot as a floating-base system. The loco-manipulation system automatically plans a full-coverage trajectory over multiple surfaces for contact-based non-destructive monitoring after unrolling the 3D-mesh region-of-interest selected from the user interface and projects it back to the surface. Our pipeline aims at offshore asset inspection and remote monitoring in industrial applications, and is also applicable in manufacturing and maintenance where area coverage is critical. We demonstrate the generality and scalability of our solution in a variety of robotic coverage path planning applications, including for multi-surface asset inspection using a quadrupedal manipulator.**

I. Introduction

Coverage path planning refers to the task of finding a trajectory (or path) that efficiently traverses a particular area while ensuring that the area is sufficiently covered. This can be a vital component for a robotic system performing tasks in a range of applications, for example, in offshore inspection [1], area monitoring [2], search and rescue [3] or landmine scanning [4]. These are scenarios where it is either difficult or dangerous for humans to work on. In the fields of agriculture and horticulture, it is beneficial to use coverage planning for agricultural machines to reduce fuel consumption by optimizing their movements during harvesting or pruning crops [5]. Examples of industry companies that develop robots with coverage techniques are Gray Matter Robotics for material sanding; and Skyline Robotics for window cleaning. Industrial tasks like surface treatment [6] tend to be repetitive and require high precision or consistent performance. Overall, coverage planning is industrially widely applicable and can be significantly more efficient and accurate when performed by robots. In field operations, where robots need to fly over or drive over a field, coverage planning requires considering either obstacles avoidance or terrain inclinations. On the other hand, with articulated robot systems covering a surface, the complication comes from the shape of a surface, robot reachability, and kinematic constraints. These features make a one-fit-all coverage planning system a challenging task to solve in robotics research.

All authors are with the Oxford Robotics Institute, University of Oxford. Their respective emails are ktien@robots.ox.ac.uk, matthew.munks@exeter.ox.ac.uk, {wolfgang,ioannis}@robots.ox.ac.uk.

(a) Coverage path (b) Physical robot setup

Fig. 1: Overview of the deployment of our proposed quadrupedal manipulator system: (a) Example of a coverage path on a reconstructed and segmented surface. (b) Physical quadrupedal robot in a multi-surface inspection scenario.

The main challenge of coverage planning is dealing with sets of complex and disjoint surfaces. For example, in real-world applications, pipes are frequently obstructed by valves, or there are many separate pipes in an inspection site requiring complex planning of inspection sequences. This highlights the need for an automated way of computing an optimized coverage path, given a list of surfaces, which can be widely applied to different applications. The optimization-based approach not only saves time and energy but also produces better solutions compared to human performance, in terms of precision and repeatability.

Intuitively, mobile robots are often preferred in large-scale surveillance, where there are many objects to be inspected, in order to achieve full coverage results. Legged robots have been widely used in unstructured environments due to their flexibility and adaptability to different terrains, and their ability to reposition their base to extend the workspace of an on-board manipulator. Therefore, they are an ideal choice for the inspection monitoring task at hand. While coverage tasks require the robot system's mobility and flexibility, we must take into account whole-body planning to guarantee reachability and stability of the robot system.

Figure 1 visualizes an example solution to coverage planning on a vertical pipe and a multi-surface task evaluation setup. In this paper, we address common issues that arise in robots performing coverage paths in remote industrial applications, which include coverage planning on disjoint surfaces, whole-body motion planning on legged manipulators, and autonomy in an unknown environment.

A. Related work

1) Robotic inspection and monitoring: Non-destructive evaluation (NDE) has been an effective approach in exam-

ining material quality in industrial or manufacturing applications such as inspection and monitoring. The advantage of NDE is that it can detect faults such as cracks, cavities and flaws without having to alter them (e.g., by drilling or cutting), which causes no disruption in service. This is especially useful in cases where regular monitoring is needed, e.g., for regular inspection of pressurized vessels used for a significant range of industrial applications. A traditional non-contact approach to non-destructive testing in manufacturing applications is visual inspection, e.g. as presented in [7]. This method can be used on top of other methods to detect visible faults on surfaces quickly, however, is not capable of detecting issues within the material or on the inside/backside of surfaces. Radiography [8] and thermography [9] are other methods that do not require contact with samples, which are applied in checking delamination in composite materials. Non-contact NDE is often fast in data collection in comparison with contact methods, hence, has its own instrumental sensitivity. Meanwhile, for NDE sensors like ultrasonic or electromagnetic, it is essential to ensure surface contact so that they can effectively collect data of the materials. An added benefit of these sensing modalities is that they can sense hidden defects that cannot be inspected through non-contact methods. An example with this approach is an electromagnetic acoustic transducer (EMAT) sensor [1], where the system automatically detects thickness change and builds a thickness map on a steel planar surface.

2) Coverage planning: Many different approaches have been developed to address coverage path planning in 2D spaces using grid-based [10], Morse-based [11] or topological coverage [12]. These methods use cellular decomposition to represent free space, then geometrically plan a path that connects adjacent cells. This task is a variant of the popular Traveling Salesman Problem (TSP) with the condition that agents should visit the neighborhood of cities. [13] introduced a boustrophedon path planner with a TSP approach for flight trajectories covering large fields. A recent survey of robotic coverage path planning is provided in [14]. While 2D coverage planning is commonly used for visual inspection [15] on Unmanned Aerial Vehicle (UAV) with onboard cameras, 3D coverage planning is necessary for contact-based monitoring.

The main challenge in robot coverage planning comes from the complexity or the large size of workspaces. Addressing reachability, [6] developed a mobile manipulator system to spray paint on pre-defined large convex surfaces. The method poses an error to the motion planner and replans if the trajectory is infeasible given the robot base pose. To inspect a large 3D structure, [15] solves a multi-UAV coverage path planning by combining Set Covering Problem and Vehicle Routing Problem. The method models coverage planning with an integer linear programming formulation to minimize the maximum length of each UAV path. Though targeting coverage planning on multiple separate surfaces, our pipeline can also generate a complete coverage trajectory on complex and large 3D objects by considering all surfaces on all sides.

3) Loco-manipulation control: One important factor of mobile manipulator inspection tasks is how to ensure that robots can strictly follow the required end-effector trajectories. Apart from planning the arm, motion planners must compute the robot base's location for achieving desired poses stably. Often, locomotion and manipulation are treated separately in controlling these models due to their complicated dynamics. Inverse reachability maps are introduced in determining base placements by analyzing the reachability and dexterity of manipulators. While collision is generally excluded in this method [16], [17], [18] use inverse Dynamic Reachability Maps [19] for optimal base placement and continuous scene monitoring and replanning to robustly accommodate dynamic changes. Sampling-based approaches [20], on the other hand, can produce collision-free motion planning in dynamic environments. Optimization is an effective tool that has been developed to calculate either base placement [6], [21] or whole-body control signals [22], [23] in loco-manipulation systems, which utilize the manipulation capabilities of mobile robots. While contact forces with the environment during manipulation tasks can cause instability in legged torsos, [24] handled such external disturbances using a trajectory optimization method and optimized whole-body loco-manipulation plans for robustness from unknown and known disturbance directions. The motion planner considers the full dynamics of the platform, therefore, can effectively plan when robot feet break contact with the environment. In [25], the authors introduced a whole-body optimization framework on a wheeled manipulator that targets task-specific constraints. The algorithm generates motion plans from the integrated inputs including state estimation, perception and users. Our work extends this concept to versatile legged mobile manipulators and includes multi-surface coverage path planning, perception, segmentation, and whole-body trajectory planning.

B. Contribution

This paper introduces a novel optimization approach to solving end-to-end loco-manipulation. Our framework integrates environment sensing, coverage path planning, whole-body trajectory planning and execution, allowing the robot to automatically perform coverage path following over multiple surfaces specified from the operator. We effectively combine 3D point cloud reconstruction and segmentation into a perception model to provide environmental awareness, which takes inputs from an RGB-D camera and a user interface. The pipeline includes a two-layer hierarchical structure of optimization: mission planning and motion planning. The former optimizes surface sequencing with a mixed-integer linear programming (MILP) model. Meanwhile, for motion planning, our work utilizes one-step and trajectory optimization algorithms. These are used to compute whole-body motion plans and joint control signals for the robot to follow the generated Cartesian trajectory. Our floating planar-base approach to the torso makes it possible for the system to be applied on either legged or wheeled mobile platforms. The full system proved to be universal and scaleable to a range of coverage path applications.

II. Problem formulation

In this section, we discuss three main components of our work: Perception, Mission planning and Motion planning.

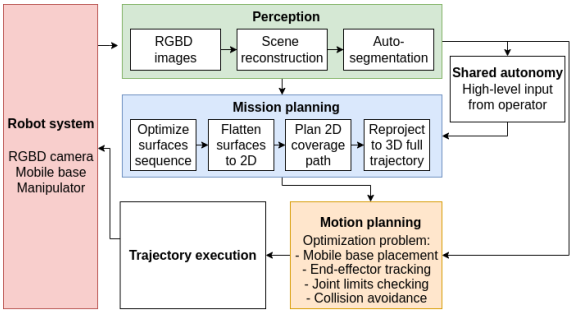


Fig. 2: Design of the proposed perceptive locomanipulation system in multi-surface inspection.

The full design is described in Fig. 2, which computes a complete trajectory over surfaces without any prior information. The goal of our work is to solve an asymptotically optimal coverage strategy over multiple random surfaces for mobile manipulators. The planning is done offline with a static environment that is unknown beforehand and is perceived on the spot. Details of each component are discussed below.

A. Perception

Our perception pipeline includes two main features: reconstruction and segmentation. The module takes input from an onboard camera and displays auto-segmented environmental objects in a user interface. Figure 3 summarizes how our perception pipeline reconstructs and segments the environment to extract visual information.

1) Reconstruction: We use an RGB-D camera on the arm's end-effector to sense the environment, which is also the only sensor used in the system. The robot is commanded to scan the scene from different angles at a fixed location. We reconstruct the environment using Truncated Signed Distance Function (TSDF) volume integration [26] with multiple input frames collected from raw RGB-D data. The method takes transformations of the camera with respect to the world coordinate to compute signed distance and utilizes viewing angles for reconstructing surface normals. The reconstructed point cloud is then downsampled with voxels to reduce processing time in later stages. Due to the noise from the camera and robot transformation during whole-body movement, we add a radius outlier filtering layer that removes all invalid points to finalize our environment point cloud. Fig. 3b shows the reconstructed point cloud from RGB-D camera scanning the environment in Fig. 3a.

2) Segmentation: To assist users in selecting a surface to scan, we segment objects from the reconstructed point cloud by combining Random Sample Consensus (RANSAC) [27] and Euclidean clustering [28]. These methods do not require any known models, which is suitable for processing unstructured environments like remote industrial sites for which no reliable prior map may exist and rapid adaptation using surface scanning is necessary. RANSAC is well-known for its plane segmentation algorithm that determines a plane model by calculating distance error with a hypothesis plane. At first, we use plane segmentation to remove the ground/table and wall from the point cloud. The objects are then clustered based on density with Euclidean algorithm, which can help to segment arbitrary shapes. Segments are then colored and

(a) Environment (b) Reconstructed (c) Segmented

Fig. 3: An example of our perception pipeline.

visualized in a user interface (Fig. 3c) and operators can either choose a segmented object or select a custom surface with a bounding box. Users are allowed to specify one or more surfaces for multi-surface planning.

B. Mission planning

1) Sequence planning: The surface sequencing is formulated as a mixed-integer linear programming problem and is an extension of integer linear programming in TSP. Taking each surface as a city in TSP, the problem is similar in the sense that we should find an optimal path through all surfaces and each surface should be visited only once. However, in our case, each surface is configured as a polygon instead of node in maps. Hence, our formulation also includes continuous variables, which are start and end positions on each surface, apart from the integer decision for surface sequencing. Our model will optimize the sequence of surfaces to be scanned in order to minimize energy. The coverage path within each surface does not affect this cost. A task sequence parameterization vector is defined for the decision, which contains the surface, start and end poses on that surface.
Equation (1) models the planning problem with $N(>=1)$ surfaces where x_{ij} is a binary decision (2) at step i of surface j, which equals to 1 if surface j is to be scanned at step i. S and G are the start and end position matrices (3). $S[,i]$ and $G[,i]$ is the Cartesian start and end positions of the robot's end-effector on the surface-to-scan at step i, which is surface j if $x_{ij} = 1$. The objective function $d()$ is the weighted Euclidean distance between end poses on the previous surface and start poses on the later surface. Constraint (4) and (5) ensures there is only one surface scanned per step and all surfaces are scanned only once. Finally, constraint (6) limits the end-effector start and end poses within the boundary of the surface at that step.

$$\min_{x,S,G} \sum_{i=1}^{N} \mathrm{d}(S[,i], G[,i-1]) \tag{1}$$

$$\text{s.t. } \forall_{i,j\in 0,..,N-1} : x_{ij} \in \{0,1\} \tag{2}$$

$$S, G = \mathbb{R}^{3\times N} \tag{3}$$

$$\forall_{i\in 0,..,N-1} : \sum_{j=0}^{N-1} x_{ij} = 1 \tag{4}$$

$$\forall_{j\in 0,..,N-1} : \sum_{i=0}^{N-1} x_{ij} = 1 \tag{5}$$

$$\forall_{x_{ij}=1} : S[,i], G[,i] \in \mathbf{B}_j \tag{6}$$

2) Coverage path planning: With each surface on the plan, the system implements the path planner from [13] to autonomously generate a coverage path. This planner processes the coverage path on a 2D surface. Therefore, we

need to flatten the 3D point cloud cropped by the Open3D [29] point selector to a polygon. The points in 3D are mapped into a flat 2D space by projecting adjacent mesh vertices into the plane of the current one [30]. We keep a reference of where the points in 2D have come from to allow for an easier transformation of the path back into 3D. The 2D polygon is fed to the coverage path planner along with the specified width of the sensor. Then, we re-project the generated coverage path into 3D with the saved reference. Randomizing positions are added to the computed 2D coverage path to ensure a smooth trajectory after re-projection.

C. Motion planning

We compute motion planning for the whole robot system, which includes both mobile base and manipulator, instead of treating them separately. This approach allows base placement generation integrated in the algorithm and maximizes dexterity in manipulation tasks. We formulate the whole-body motion planning as a nonlinear optimization problem using the Extensible Optimization Toolkit (EXOTica) [31]. The planning scene is defined with a complete loco-manipulation robot model, where the torso is treated as a floating-base system. In our work, the mobile platform is holonomic and does not have any restrictions on translation or rotation, however, these could be added if required. The state of the robot with N-DoF manipulator is described as:

$$x = [q_{base}, q_{manipulator}] \tag{7}$$

$$\text{where } q_{base} = [x_{base}, y_{base}, yaw_{base}] \tag{8}$$

$$q_{manipulator} = [q_1, q_2, q_3, ..., q_N] \tag{9}$$

In this study, we compare two methods of optimization: one-step and trajectory. Details of the algorithms are discussed below.

1) One-step optimization: The whole-body one-step optimization is defined as a quadratic cost problem:

$$\arg\min_{x} f(x)^T Q f(x) \tag{10}$$

with cost function:

$$f(x) = \sum_i \rho_i \left\| \Phi_i(x) - y_i^* \right\| \tag{11}$$

where ρ_i is the weight of task i, Φ_i is the mapping function of the task map and y_i^* is the goal reference. The robot is asked to reach desired arm's end-effector position and orientation in the task space generated by the path planning module. The 6-DoF robot arm is reduced to a 5-DoF system where the final joint is ignored since this yaw value does not affect the arm end-effector tracking. We employ the generalized inverse kinematics (IK) solver to compute joint position signals with minimized error with the reference poses, i.e. a multi-objective optimization over a desired set of tasks. Joint limits are checked using the robot model definition in the planning scene. This algorithm computes a single configuration for robot joints at each step on the trajectory and continues to the next stage until the whole trajectory is completed. The optimization loop continuously updates the current state of the robot as the initial state for the next step's optimization problem to avoid local minima and smooth the robot movement. The cost of the optimization problem is used to automatically enable base position constraint, aiming at minimizing base movement and improving scanning time. Accordingly, if the cost is below a pre-defined threshold, the problem will transform to a fixed-base manipulator reaching desired end-effector poses task.

2) Trajectory optimization: For the trajectory optimization, we model the whole-body control for a legged mobile manipulator as a one-step look-ahead nonlinear optimization in time configuration space with constraints:

$$\arg\min_{x,u} f(x, u, t), \tag{12}$$

$$\text{s.t. } h(x, u, t) = 0, \tag{13}$$

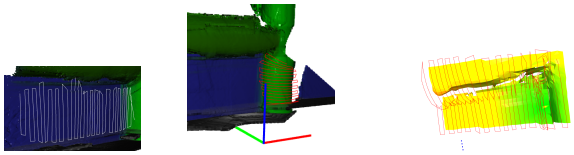
$$g(x, u, t) \leq 0, \tag{14}$$

where u is the control signal. Given the desired end-effector trajectory, equality constraints $h(x, u, t)$ are set as the arm's end-effector position and orientation goal. Meanwhile, inequality constraints $g(x, u, t)$ place limits on joint position and velocity. The trajectory optimization method plans the control signals, considering the next state on the computed path, which ensures smooth movement between steps. We also penalize base movement along the trajectory. We use variable number of timesteps for the trajectory optimization, which is found in the computed path from the path planning module. Instead of generating separate joint positions for each step, this algorithm computes a complete time-indexed joint acceleration trajectory. We use second-order derivatives on cost functions to obtain smooth transitions during horizon planning. A Differential Dynamic Programming (DDP) solver is applied to solve the trajectory optimization, which is efficient in handling timing constraints in the control loop. As the problem is highly nonlinear, this solver is not guaranteed to converge and needs a specified maximum iteration. We relax the constraints during approaching the trajectory to allow the robot more time to settle in the initial step and follow the path afterwards.

III. Evaluation

A. Single-surface planning

This section validates the coverage path planning adaptability on different surface types and their coverage quality. Fig. 4 captures three experiments to validate our work of planning coverage paths on random surfaces. The distance from the surface to the paths is adjustable, which allows different applications for monitoring with visual sensors as well as other, e.g. contact-based, modalities.



(a) Flat surface (b) Cylinder (c) Non-flat surface

Fig. 4: Coverage path planning results on different surfaces. The surfaces have been automatically reconstructed and captured using our method prior to coverage path planning.

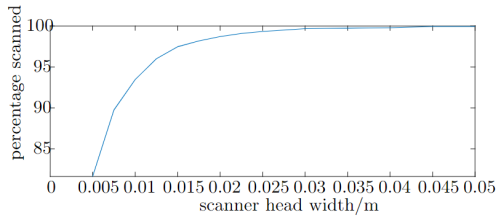


Fig. 5: Coverage results of the path planning algorithm over sensor head width on a flat surface.

In all three experiments, flat contours that are fairly closely aligned are generated. The regions were selected by the user to inspect coverage on different parts. Figures 4a and 4b show the system working on flat and curved surfaces respectively. The path around the vertical cylinder is slightly distorted at the top but this will still allow full coverage. Figure 4c shows an attempt of path planning over such a surface (where the corner formed by the two cylinders and the rear surface is the problematic area). There is a large amount of distortion at that corner, but the algorithm performs better than expected across the remaining portion of the surface.

We evaluated the coverage percentage of the path planning module by looking at adjacent contours in sequence. For each pair of contours, a triangular mesh is generated. This enables us to calculate both the total area to scan, which is the sum of the areas of all the triangles, and the area that the scanner passes over per triangle, which is a trapezium at the bottom and approximated as a triangle at the apex of the triangle. Despite this approximation, it should be suitable for the case where the contours are roughly parallel to each other, which is the case here.

Fig. 5 plots the percentage of the chosen surface scanned against the scanner head width. The contours in these tests are all supposedly $0.03\,\mathrm{m}$ away from each other, i.e. the size of each pixel in 2D space. We can see that the algorithm produces a very high percentage coverage for this width and those above it, as well as doing well on those slightly below it. As the scan quality decreases, the percentage of the surface that is not covered increases. This is due to small kinks in the path, and fewer points at the edges of each scan. This stops the edges of the scan from properly aligning, thus creating a few imperfections in the edges.

Note that Fig. 5 was done using a set of passes over a flat surface, as shown in Fig. 4a. This produced a coverage percentage of $99.6\,\%$. For the cylinders in Figure 4b, we get $96.0\,\%$ coverage. This is due to a slightly higher variance in the spacing of the contours themselves, which in itself is less due to the curvature and has more to do with the length of each contour over the surface. With this percentage of coverage, any flaws in the material that are of finite size are likely to be detected by the system.

We also deployed our perception pipeline with data collected from a physical environment. Fig. 6 shows that our work can successfully reconstruct, segment and plan a coverage path on noisy data in the real world.

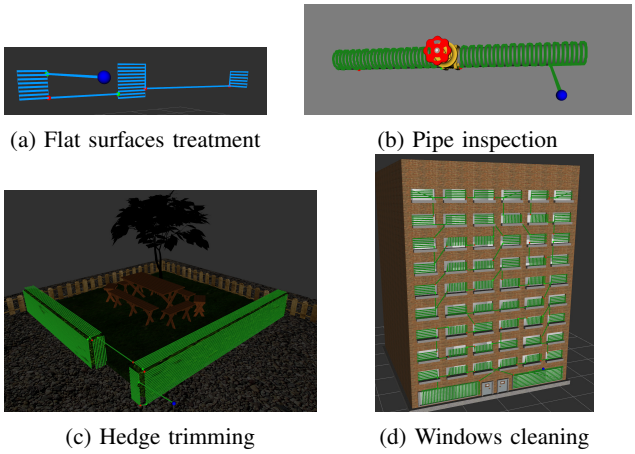
B. *Multi-surface mission planning*

A set of scenarios are selected to show how our mission planning is universal to many coverage planning applications,

(a) Real world (b) Auto-segmented (c) Coverage path

Fig. 6: Perceptive module tested on real environmental data.

as visualized in Fig. 7. In these figures, the blue dot is the current robot's end-effector position while green and red ones represent the computed start and end positions on surfaces. The complete trajectory in Cartesian space, which includes both the surface sequence and coverage paths on all surfaces, is denoted by the blue/green line. This path is the output of the mission planning optimization pipeline. We use Gurobi [32] to solve the optimization problem.



(a) Flat surfaces treatment (b) Pipe inspection

(c) Hedge trimming (d) Windows cleaning

Fig. 7: Set of experiments highlighting applicability of our developed method to a variety of multi-surface coverage path planning tasks.

Fig. 7a describes a simple scenario with three separate flat surfaces, which is applicable in material sanding or wall painting. Meanwhile, Fig. 7b shows that our pipeline can be used where the coverage planning is obstructed by a valve and has to follow a cylindrical surface posing challenges for whole-body motion planning, which is typical in pipe inspection tasks. These are different types of 3D surfaces that our system can plan coverage on. Fig. 7c is a hedge-trimming scenario where coverage paths can be planned in different directions. This is an example where the coverage path has to scale to a large 3D structure. Our mobile legged-manipulator system is also suitable for this task as the hedge can be on rough or rocky terrains. Finally, Fig. 7d demonstrates the scalability of the mission planning framework with 56 surfaces in a window cleaning task. Our framework can be extended to consider penalizing motions in different directions (up/down against left/right) for real-world objective functions – even though our legged robot system is not suitable for this scenario, the developed method applies to any floating-base manipulation system, e.g. similar to Skyline Robotics.Our set of experiments can serve as a benchmark for measuring multi-surface coverage path planning for loco-manipulation.

TABLE I: Evaluation of multi-surface mission planning.

Number of surfaces	Planning time (s)	Optimality gap (%)
2	0.021	0
3	0.007	0
5	0.015	0
10	0.23	0
15	16.856	0
20	1000	38.86
25	1000	68.7
30	1000	74.73

Table I shows the results of our multi-surface mission planning over the number of surfaces. We set time limit 1000 s for the planning time. The optimality gap is the expected gap to optimal solution, which is computed by:

$$\frac{|\text{ObjectiveBound} - \text{IncumbentObjectiveValue}|}{|\text{IncumbentObjectiveValue}|}$$

The result shows that our pipeline can obtain globally optimal plans on up to 15 surfaces within 17 s. Longer time limits should be taken to reach optimality with larger state spaces, e.g. the 56-surface window cleaning in Fig. 7d.

C. Pipeline integration

We validated our fully integrated system with one-step and trajectory optimization, and provide a comparison between the two approaches. The mobile manipulator used in both experiments is a lightweight 6-DOF Kinova robot arm mounted on an Anymal C quadruped, as shown in Fig. 1b. We attach an Intel Realsense D435 depth camera to the manipulator's wrist for scene reconstruction and visual sensing.

The testing scenario is a multi-surface inspection task where the robot is asked to scan a region automatically with input surface from the operator. The full pipeline of our system includes the following steps:

1) Scan the surface from a location with different angles.
2) Process collected RGB-D images: reconstruct the surface, filter noises, segment objects.
3) Ask the operator to select one or more objects or custom surfaces to scan.
4) Plan the full coverage path over all selected surfaces.
5) Solve the whole-body optimization problem to get joint positions control signal.
6) Execute the joint trajectory.

Aiming at comparing performance of the two optimization methods, steps 1-4 remain the same while we switch between the one-step and the trajectory mode. The planned coverage trajectory from step 4 is automatically discretized to a number of desired poses midway. The one-step optimization plans control signals for each step on the path. Meanwhile, trajectory planning method defines time-steps needed to perform each pose on the path.

The robot first moves to the desired base pose, then the manipulator is controlled to reach waypoint target positions and orientations accordingly.

In Fig. 8, we compare task costs, which include position, angle alignment and joint limit violation, over steps in the two optimization methods with the same trajectory. All task weights are set the same respectively in both modes. The results show that it takes time for the trajectory planning method to settle on the coverage path, but have better

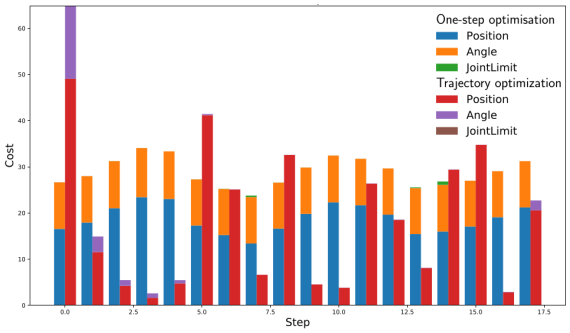


Fig. 8: Detailed task cost comparison between one-step and trajectory optimization.

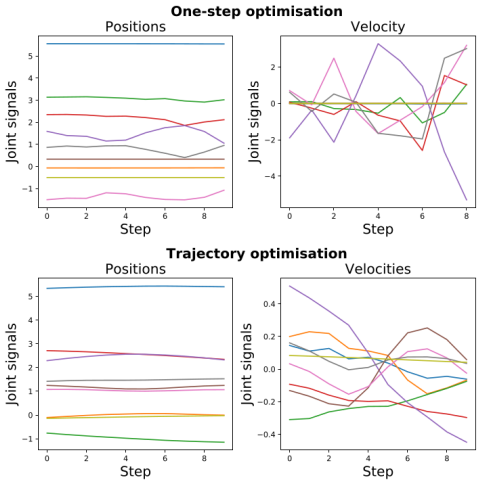
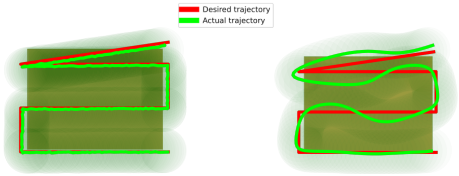


Fig. 9: Joint states comparison between one-step and trajectory optimization.

alignment with significantly lower cost at some poses in later steps. Notably, no joint limits are violated in the trajectory optimization. In general, the one-step approach produces more consistent costs along the coverage path in comparison with trajectory optimization. This suggests that separating approach timing from trajectory following could lead to optimal solutions.

Fig. 9 depicts the joint states computed from the one-step and trajectory optimization over timesteps. The 9 lines in each subplot represent control signals for our 9-DOF system (3-DOF signal for the base and the rest for the arm). It is clear that joint positions are generally stable in both modes. However, the one-step optimization has sightly more fluctuation in these signals compared to the trajectory approach. In conclusion, the trajectory planning algorithm generates a smoother movement with significantly lower velocity.



(a) One-step optimization (b) Trajectory optimization

Fig. 10: Coverage result of our full pipeline.

Fig. 10 visualizes the coverage path tracking on both

motion planning methods. The red lines refer to the generated trajectory from the mission planning module while the green lines denote the performed trajectory from the full pipeline integration. The inspected area is represented by the green faded padding, where the overlapped region gets a darker shade. The radius is the sensor head width sent to the coverage path planning system. In one-step optimization path tracking (Fig. 10a), the performed trajectory is almost the same as the generated one, with its padding proving that the generated trajectory ensures full coverage over the inspected region. The results show that our system successfully follows the coverage trajectory and the pipeline efficiently minimizes overlapping, which optimizes energy in inspection tasks. In the plots, there are observed small gaps that are not covered due to the absolute radius (without overlapping) that we set in the coverage path planning, which can be decreased to compensate errors from robot motion planning and control.

Supplementary material and videos on our experimental evaluation are available at `https://sites.google.com/view/multisurface-coverage-planning`

IV. Discussion

We have proposed an end-to-end optimization-based framework for loco-manipulation systems to perform coverage path planning in inspection and monitoring applications. Our main contribution is the introduction of a two-layer optimization structure for the fully integrated system, where the coverage mission planning also optimizes the start and end positions on the surfaces. Additionally, the perception system automatically reconstructs and segments the environment scene, which intuitively helps users in selecting the inspected surface. Along with that, we developed a full-coverage path planning method that successfully unrolls 3D meshes from a point cloud, generates a coverage 2D path and projects it back to the surface. The pipeline can be universally applied on multiple scenarios and can scale to large state spaces with longer processing time. The set of surface sequencing experiments can also be used as a benchmark for multi-surface coverage planning. Two optimization techniques were evaluated for motion planning, one-step and trajectory optimization planning. Both approaches consider robot reachability, alignment, collision avoidance and capability of base relocation. The results showed that the trajectory optimization produces a smoother movement but with less consistency in poses alignment than the one-step approach. Therefore, the one-step motion solution was preferred if strict requirement of path following (e.g. contact-based applications) was in place. Though, better tuning of trajectory optimization might help this method to produce similar or better alignment with a lower overall cost.

In the future, we would like to integrate the path cost from the coverage path planner into the mixed-integer mission planning objective function. The problem could then be solved with a black-box optimization solver, e.g., [33]. In addition, the proposed framework currently assumes that motion planning can always find a feasible solution that follows the computed Cartesian path resulting from mission planning. Therefore, further development can also involve informing surface sequencing with geometry feasibility and reachability evaluation.

References

[1] V. Ivan, *et al.*, "Autonomous Non-Destructive Remote Robotic Inspection of Offshore Assets," in *Offshore Technology Conference*, 2020.

[2] S. M. Ahmadi, *et al.*, "Constrained coverage path planning: evolutionary and classical approaches," *Robotica*, vol. 36, no. 6, 2018.

[3] D. Jia, *et al.*, "Coverage path planning for legged robots in unknown environments," in *IEEE SSRR*, 2016.

[4] D. Khudher, "Continuous Landmines Scanning using Legged Robot with Manipulator Arm over Rough Terrain," in *IEEE AIM*, 2017.

[5] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *J. of Intelligent & Robotic Systems*, vol. 74, no. 3-4, 2014.

[6] N. Dhanaraj, *et al.*, "A Mobile Manipulator System for Accurate and Efficient Spraying on Large Surfaces," in *Int. Conf. on Industry 4.0 and Smart Manufacturing*, 2022.

[7] J. E. See, *et al.*, "The Role of Visual Inspection in the 21st Century," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 61, no. 1, pp. 262–266, 2017.

[8] C. D. Lockard, "Anomaly detection in radiographic images of composite materials via crosshatch regression," Ph.D. dissertation, Mills College, 2015.

[9] C. Maierhofer, *et al.*, "Characterizing damage in CFRP structures using flash thermography in reflection and transmission configurations," *Composites Part B: Engineering*, vol. 57, pp. 35–46, 2014.

[10] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, 2001.

[11] E. U. Acar, *et al.*, "Morse decompositions for coverage tasks," *IJRR*, vol. 21, no. 4, pp. 331–344, 2002.

[12] S. Wong and B. MacDonald, "A topological coverage algorithm for mobile robots," in *IEEE IROS*, 2003.

[13] R. Bähnemann, *et al.*, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," in *Field and service robotics*. Springer, 2021, pp. 277–290.

[14] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[15] W. Jing, *et al.*, "Multi-uav coverage path planning for the inspection of large and complex structures," in *IEEE IROS*, 2020.

[16] A. Makhal and A. Goins, "Reuleaux: Robot Base Placement by Reachability Analysis," in *IEEE IRC*, 01 2018, pp. 137–142.

[17] J. Dong and J. C. Trinkle, "Orientation-based reachability map for robot base placement," in *IEEE IROS*, 2015.

[18] W. Merkt, *et al.*, "Robust shared autonomy for mobile manipulation with continuous scene monitoring," in *IEEE CASE*, 2017.

[19] Y. Yang, *et al.*, "iDRM: Humanoid motion planning with realtime end-pose selection in complex environments," in *IEEE Humanoids*, 2016.

[20] ——, "Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints," in *IEEE Humanoids*, 2018.

[21] Q. Yu, *et al.*, "Base position optimization for mobile painting robot manipulators with multiple constraints," *Robotics and Computer-Integrated Manufacturing*, vol. 54, pp. 56–64, 2018.

[22] C. D. Bellicoso, *et al.*, "ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot," in *IEEE ICRA*, 2019.

[23] P. Ewen, *et al.*, "Generating Continuous Motion and Force Plans in Real-Time for Legged Mobile Manipulation," in *IEEE ICRA*, 2021.

[24] H. Ferrolho, *et al.*, "RoLoMa: Robust Loco-Manipulation for Quadruped Robots with Arms," *arXiv:2203.01446*, 2022.

[25] W. Merkt, *et al.*, "Towards shared autonomy applications using whole-body control formulations of locomanipulation," in *IEEE CASE*, 2019.

[26] Q.-Y. Zhou and V. Koltun, "Dense Scene Reconstruction with Points of Interest," *ACM Transactions on Graphics*, vol. 32, 07 2013.

[27] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, jun 1981.

[28] M. Ester, *et al.*, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Int. Conf. on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[29] Q.-Y. Zhou, *et al.*, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847*, 2018.

[30] L. Fraser, "How to flatten 3D object surface into 2D array?" 2017.

[31] V. Ivan, *et al.*, *EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control*. Springer, 2019.

[32] B. Bixby, "The Gurobi optimizer," *Transp. Research Part B*, 2007.

[33] R. Hamano, *et al.*, "CMA-ES with Margin: Lower-Bounding Marginal Probability for Mixed-Integer Black-Box Optimization," *arXiv:2205.13482*, 2022.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

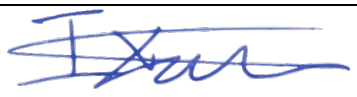
Title of Paper	Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Kim Tien Ly, Matthew Munks, Wolfgang Merkt, and Ioannis Havoutis. "Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring". *Proc. IEEE International Conference on Automation Science and Engineering (CASE)*, 2023.

Student Confirmation

Student Name:	Kim Tien Ly		
Contribution to the Paper	• Developed the project idea • Wrote the code for the pipeline • Conducted the experiments • Collected data and performed analysis • Wrote large portions of the paper		
Signature		Date	04/10/2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Ioannis Havoutis		
Supervisor comments		
Signature 	Date	07/10/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

Addendum

I would like to add further explanation to the paper "Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring".

Gurobi solver [96] determines optimality with the metric optimality gap (Table I), which measures how close the current solution is to the optimal solution. Accordingly, optimality gap is computed by:

$$\frac{|\text{ObjectiveBound} - \text{IncumbentObjectiveValue}|}{|\text{IncumbentObjectiveValue}|},$$

where

- ObjectiveBound is the best known bound on the optimal objective. When solving a model, the algorithm maintains both a lower bound and an upper bound on the optimal objective value. For a minimization model like ours, the upper bound is the objective of the best known feasible solution, while the lower bound gives a bound on the best possible objective.
- IncumbentObjectiveValue is the objective value for the current solution. If the model was solved to optimality, then this attribute gives the optimal objective value.

4.2 Discussion

This chapter presented an end-to-end TAMP solution for optimizing coverage path planning for robotic systems in industrial asset inspections. The framework is a two-layer hierarchical system that combines mission planning and motion planning, allowing for automatic planning of a full-coverage trajectory on multiple surfaces in unknown environments.

For mission planning, we designed a MILP formulation that solves both the sequence of surfaces and the start and end poses on each surface. We coupled this multi-surface sequence with a local coverage path planner on individual surfaces to complete a Cartesian trajectory as the output of the high-level planner. Motion planner is the second layer that optimises a whole-body solution in configuration space on a floating-based manipulator in order to control the robot following the generated end-effector path. We investigated the difference between one-step and trajectory optimization and discussed the suitable applications for each method. Our floating planar-base method for the torso allows the system to be adaptable to both legged and wheeled mobile platforms.

The experimental results evaluate each level of the hierarchical structure and the overall coverage outcome. Evaluation results show the system's generality and adaptability, demonstrated through experiments on surfaces with shape variants. The mission planning pipeline efficiently sequenced the order of surface inspections and optimized the robot's movement across different surfaces. We validated the scalability of the system to different numbers of surfaces across different scenarios. For motion planning, one-step optimization consistently produced lower task costs across steps, resulting in better path alignment and precise coverage in scenarios where exact path following is crucial. On the other hand, trajectory optimization provided smoother joint movements and lower velocities, which is beneficial for tasks requiring fluid motion, but it exhibited slight inconsistencies in path alignment. Additionally, the perception pipeline sufficiently reconstructs and segments the environment that assists the users in teleoperation situations. The integration of

these components into the full pipeline ensured that the robot could autonomously execute complex inspection tasks and perform full coverage on mobile manipulators.

In conclusion, the proposed framework demonstrates advancements in loco-manipulation coverage path planning, especially in enhancing the robustness of multi-surface inspection scenarios. It offers a scalable, optimization-oriented solution on mobile manipulators, with potential applications extending across multiple industries.

Lastly, the one-way structure of our hierarchical system assumes that the robot can find a motion plan for the generated Cartesian coverage path from the mission planner. There is no re-planning or feasibility check capability during high-level planning to ensure executability, assuming camera scanning only covers reachable regions. This motivates us to develop later projects, where task planning and motion planning are more tightly integrated with reachability-aware planning. Furthermore, this work investigates multi-surface coverage problems with mobile manipulators in industrial settings, which is tightly aligned with this problem. We explore more general-purpose robotic systems in the later chapters.

5
Reachability-guided Optimal TAMP

This chapter addresses a tightly integrated TAMP solution, where the symbolic planner and the motion planner interact effectively to obtain robust embodied intelligence. In the hierarchical optimization-based framework developed in Chapter 4, the optima of each layer might not be the overall optimal solution. Logic-geometric programming (LGP) [3] interestingly formulated the hybrid TAMP problem in one optimization formulation, which motivates the development of this chapter. Specifically, this project explore a TAMP approach with the following aspects

Task specification	PDDL
Interaction between task and motion planning layers	Integrated
Optimality vs feasibility	Optimal
Open-loop vs closed-loop	Open-loop
Role of human	None

One of the limitations of LGP is that it does not scale well to high-dimensional systems and can suffer from obstacle avoidance issues. A major cause of those failures is that high-level strategies might not be aware of the geometric and kinematic robot-centric constraints, which limit the effectiveness of task planning. Although a heuristic solution [75] was introduced to incorporate robot constraints, it is simplified with joint length and cannot capture the dexterity of high-DoF manipulators.

The problem becomes more complex with long-horizon tasks or multiple–possibly dynamic–obstacles. Moreover, trajectory optimization with keyframes initialization appeared to fall into the typical local minima problem of optimization methods.

The contribution of this work is the introduction of a sampling-based reachability graph that is tightly integrated with LGP to enable solving optimal TAMP on high-DoF mobile manipulators.

- The proposed graph can incorporate environmental and robot kinematics information to provide the planner with sufficient geometric constraints.
- This reachability-aware heuristic efficiently prunes infeasible sequences of actions in the continuous domain, hence, it reduces replanning by securing the feasibility of the generated plan.
- The reachability graph also provides path guidance to the final layer of LGP - trajectory optimization - to avoid local minima while encountering obstacles.

5.1 R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators

The article in this section was presented at the *IEEE International Conference on Robotics and Automation (ICRA)* 2024 [136]. Additional videos demonstrating simulated and real-robot experiments are available online at https://youtu.be/NEVVHEhQnOQ.

R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators

Kim Tien Ly[1], Valeriy Semenov[1], Mattia Risiglione[2], Wolfgang Merkt[1], Ioannis Havoutis[1]

***Abstract*— This paper presents an optimization-based solution to task and motion planning (TAMP) on mobile manipulators. Logic-geometric programming (LGP) has shown promising capabilities for optimally dealing with hybrid TAMP problems that involve abstract and geometric constraints. However, LGP does not scale well to high-dimensional systems (e.g. mobile manipulators) and can suffer from obstacle avoidance issues due to local minima. In this work, we extend LGP with a sampling-based reachability graph to enable solving optimal TAMP on high-DoF mobile manipulators. The proposed reachability graph can incorporate environmental information (obstacles) to provide the planner with sufficient geometric constraints. This reachability-aware heuristic efficiently prunes infeasible sequences of actions in the continuous domain, hence, it reduces replanning by securing feasibility at the final full path trajectory optimization. Our framework proves to be time-efficient in computing optimal and collision-free solutions, while outperforming the current state of the art on metrics of success rate, planning time, path length and number of steps. We validate our framework on the physical Toyota HSR robot and report comparisons on a series of mobile manipulation tasks of increasing difficulty. Videos of the experiments are available here.**

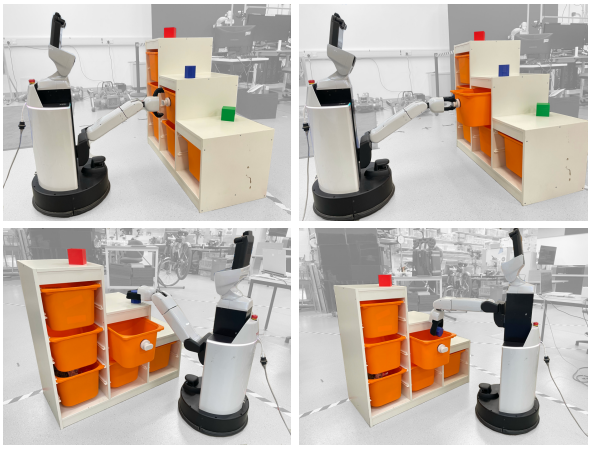


Fig. 1: Physical HSR performing R-LGP solution for table clearing task. The captured actions include grabbing knob, opening/closing drawer, picking object, and dropping object.

I. Introduction

Task and motion planning (TAMP) is the process of decision making that takes a given task as input and outputs a sequence of robot configurations to complete the task. While task planning focuses on high-level task-oriented strategies, motion planning refers to low-level control algorithms that determine the feasibility of robot motion and the continuity considering actuation and joint limits, environmental obstacles, or uncertainties. These methods take into account the variations of sequences of actions that can lead to the desired goal. Therefore, TAMP planners must address both task-level and motion-level requirements, which can be solved either independently or jointly.

A major challenge in TAMP is that high-dimensional geometric constraints in motion control (kinematics, joint limits, reachability, etc.) can limit the effectiveness of high-level strategies. The problem becomes more complex with long-horizon tasks, higher-degrees-of-freedom robots, or multiple–possibly dynamic–obstacles. Therefore, the deliberative function (high-level task planner) should either be informed or have sufficient restrictions on the robot's physical limitations to produce feasible solutions. Consequently, TAMP requires the integration of both high- and low-level planning and TAMP research generally aims to effectively combine both artificial intelligence techniques in task planning and advanced motion planning to tackle such problems.

In this paper we propose a novel TAMP approach to solve sequential decision-making applications inheriting logic-geometric programming (LGP) [1]. Our framework generates optimal plans for complex robot tasks, in contrast to current approaches that generate solely feasible solutions, for example, the well-known TAMP solver PDDLStream [2]. Our approach tightly integrates a reachability graph with LGP and allows us to achieve robust, collision-free and kinematically-efficient solutions to LGP on high-dimensional mobile manipulators. We explore the efficacy of our framework in a range of existing TAMP problems of increasing difficulty and report results on success rates, planning times and path lengths. Furthermore, we validated on a physical robot, the Toyota Human Support Robot (HSR), aiming at realistic and practical applications, where geometry and kinematics must be respected.

Our main contributions are (1) a LazyPRM-inspired graph for motion planning on floating-base manipulators, (2) a reachability graph serving as symbolic and motion guidance for the LGP planner, (3) a kinematically-effective optimization-based system for TAMP on mobile manipulators. Our novel TAMP approach is a Reachability-guided Logic-geometric Programming framework, in short, *R-LGP*.

[1]K. T. Ly, V. Semenov, W. Merkt and I. Havoutis are with the Dynamic Robot Systems (DRS) group, Oxford Robotics Institute, University of Oxford. Email: {ktien,valeriy,wolfgang,ioannis}@robots.ox.ac.uk.

[2]M. Risiglione is with Istituto Italiano di Tecnologia. Email: mattia.risiglione@iit.it.

II. Related work

A. Satisfactory TAMP

The main challenge in combining task and motion planning is their hybrid nature. While task planning involves discrete task specifications, motion planning is typically solved in continuous space. Task and motion planning are traditionally combined on a level basis, which can be decoupled or integrated. Accordingly, each layer will keep its own level of abstraction and the TAMP model should be able to translate knowledge between the two levels. Given the introduction of STRIPS (Stanford Research Institute Problem Solver) [3], robot planning was initially studied with a decoupled hierarchical structure [4]. The approach in this early work assumes that motion control can solve all high-level actions without alternative backtrack solutions. Meanwhile, integrated TAMP [5] considers failures and can replan to ensure that the strategy is executable. Common methods are taking one of the two parts as the base plan and solving the other level accordingly. For example, Srivastava et al. [6] introduces a TAMP method that backtracks and finds alternative task plans if the motion solver fails. Such approach usually discretizes the continuous space for symbolic planning. The method can utilize off-the-shelf planners and state-of-the-art techniques. TMKit [7] is introduced as the first open-source framework to implement such a structure, based on [8][9]. There is an opposite approach where the task search space is bounded by sampling the continuous workspace [10]. Recently, Kim et al. [11] introduced a reachability tree-based sampling algorithm that pre-generates goal state to bias action search. Given that task planning is usually in a finite domain, having to re-plan the task sequences is considered to produce a lower cost compared to continuously checking geometrical feasibility.

PDDLStream [2] extends Planning Domain Definition Language (PDDL)[12] by using streams to enable sampling procedures. This work proves to be a modular and domain-independent approach to TAMP.

B. Optimal TAMP

To reduce complexity, TAMP approaches usually tackle task and motion planning separately and focus more on completeness. Solving such problems optimally is difficult because the optima of each layer might not be the overall optimal solution. An example can be found in [13], where the authors used a two-layer hierarchical structure of optimization to find asymptotically optimal solutions to coverage planning task. Kongming [14] was the first approach to integrate continuous control variables in an activity planner. The method combines a Planning Graph for discrete actions and Flow Tubes for continuous actions, which was modeled as a mixed logic linear (non-linear) program. The method ties to a fixed time discretization, which is difficult to apply on long-horizon tasks. However, it brings up the need for a tightly coupled task and motion planning framework in robotics applications.

Toussaint [1] introduced the logic-geometric programming (LGP) algorithm, which formulates TAMP as a mathematical program that uses optimization methods to find locally optimal plans instead of solely feasible ones. The task in this work is to build the highest stable tower from a list of blocks and boards. By bringing logic into geometry, LGP does not need to arbitrarily discretize the continuous space and directly optimize continuous solutions. An extended work from the team [15] integrated RRT to reconstruct a large pavilion. Although the method solves the issue when the optimizer fails in a cluttered environment, the system ignores system uncertainties. Variants of LGP papers include a heuristics method for long-horizon tasks [16][17], a dynamic LGP for human motion prediction [18] or an approximation solver for cooperative manipulation [19]. Other optimization-based mathematical approaches to TAMP is constraint programming (CP) [20] [21] and mixed integer programming (MIP) [22] [21] [23]. CP is a more general term than MIP, Booth et al. [21] did a comparison between MIP and CP and concluded that inference-based CP is better than relaxation-based MIP in terms of time and solution quality. This work is tested on simulation and claims to be case-sensitive. MIP is originally a decision making and scheduling method that solves non-convex problems. When it comes to robotics planning, this optimization solution can help to capture discrete decisions with integer variables. Deits et al. [24] introduced a planner that uses MIP to generate globally optimal sequences of footsteps in difficult terrain. Multi-robot task planning is a common application of MIP [25][26]. In short, these mathematical techniques allow encoding logical decisions and geometric constraints in nonlinear optimization models without backtracking, targeting globally optimal strategies.

As more interest has been driven to solving TAMP optimally, besides mathematical programming, a recent work from Thomason et al. [27] proposes asymptotically optimal decision-making using informed tree search. The model effectively combines constraint-based symbolic planning, distance-based predicate representation, and batch-sampling-based optimal motion planning to solve a hybrid state space problem. Recently, inspired by LGP, Sleiman et al. [28] introduced an offline bilevel optimization planner that solves multi-contact problems on loco-manipulation system. The system successfully plans whole-body motion for a quadrupedal mobile manipulator to open/close doors and dishwashers.

III. Proposed framework overview

In LGP, the authors solve TAMP with three levels of approximation and switches between symbolic search and configuration optimization that is conditioned by symbolic decisions. While *level 1* decides the symbolic sequence, *levels 2* and *3* optimize keyframes (e.g. grasp poses) and full path respectively. An extended work, RHH-LGP [16], proposes a heuristic to guide the symbolic search in level 1. The kinematic reachability here, which is an important factor to the feasibility of the action sequences, is fixed or calculated using Euclidean distance or the length of robot's links. In this work, we extend LGP with a sampling-based reachability graph to enable solving optimal TAMP on high degrees-of-freedom (DoF) mobile manipulators. The proposed reachability graph can also incorporate environmental information (obstacles) to provide the planner with sufficient geometric constraints. This reachability-aware heuristic efficiently prunes infeasible sequences of actions

Algorithm 1 Reachability-guided LGP

Require: symbolic goal g, initial symbolic state s_0, initial configuration x_0, world W

$s \leftarrow s_0$
while not $s \in S_{goal}(g)$ **do**
  PathFound $\leftarrow$ False
  SymbolicSearch $\leftarrow s$
  Switch $\leftarrow \emptyset$
  Path $\leftarrow \emptyset$
  while not PathFound **do**
    // Construct reachability graph
    RG = constructRG(W)
    // Symbolic search
    $n \leftarrow$ SymbolicSearch.argmin(heuristicCost(**RG**))
    Switch.append(n)
    if not $n \in S_{goal}(g)$ **then**
      SymbolicSearch.append(n.expand())
    else
      // Optimize over kinematic switches
      if SwitchOptimization(Switch) **then**
        waypoints $\leftarrow$ getWaypoints(**RG**, n)
        Path.append(waypoints)
        // Optimize over the full path
        if PathOptimization(Path) **then**
          PathFound $\leftarrow$ True
          $s \leftarrow n$
      else
        break

in the continuous domain. Hence, it reduces replanning by securing feasibility at the final full trajectory optimization.

Our reachability-guided LGP (R-LGP) pipeline is described in Algorithm 1. The proposed reachability graph serves as a guide to the first (symbolic) and final (full path) layers in LGP. In the pipeline, nodes chosen from the symbolic search, with the help of our reachability heuristic, will be sent to the kinematic switch optimization. Once a feasible sequence is found, the system will solve the full path optimization with the guided path from the precomputed reachability graph. The main contribution of the graph is twofold: *a)* to provide a heuristic that informs the LGP's symbolic planner about kinematics and geometry; *b)* to provide collision-free guidance to the LGP's trajectory optimization at level 3 (full path), in order to avoid local optima.

IV. Reachability Graph

The reachability graph is designed in a LazyPRM-like [29] manner. We compute the graph with node validation and edge checking during planning.

A. Graph generation

This section explains the **constructRG** function in Algorithm 1.

1) Node sampling: Nodes are randomized in task space $x \in \mathbb{R}^3$ with uniform distribution $U(p_{\min}, p_{\max})$. Nodes in the graph represent end-effector positions of the mobile manipulator, which sufficiently denotes robot reachability. We validate nodes with a constrained optimization formulation. The robot model is defined as a loco-manipulation system with a floating-base torso as in Eq. 1, where m is the number of DoFs of the manipulator. This allows the framework to be implemented on either legged or wheeled mobile platforms. In our work, the mobile platform is holonomic and does not have any restrictions on translation or rotation, however, these can be added in as constraints.

$$\begin{aligned} q &= [q_{\text{base}}, q_{\text{manipulator}}], \\ \text{where } q_{\text{base}} &= [x_{\text{base}}, y_{\text{base}}, \psi_{\text{base}}], \\ q_{\text{manipulator}} &= [q_1, q_2, q_3, ..., q_m]. \end{aligned} \tag{1}$$

The nonlinear optimization formulation for node validation is defined in Eq. 2, where x_0 is the checked node, representing the reference for the robot end-effector position. $g_{\text{prec}}()$ checks collision between the inspecting node and the environment, and determines the precondition for the optimization with a constant $M \rightarrow +\infty$. This condition removes unreachable nodes for the sake of processing time. On the other hand, $f_{\text{kin}}()$ computes the end-effector position x that corresponds to the joint values q. Function $g()$ defines inequality constraints for whole-body joint limits and collision. In the reachability graph, orientation of the end-effector is not constrained for flexibility and generality. Our graph is designed to generate a guiding cost and path for LGP. The full trajectory optimization, with orientation constraints, will then be computed on the final layer with constraints for the relevant manipulation action.

$$\begin{aligned} \min_{q} \; & M \cdot \max(0, g_{\text{prec}}(x_0)) + |x - x_0|^2 \\ \text{s.t. } & g_{\text{prec}}(x_0) \leq 0, \\ & x = f_{\text{kin}}(q), \\ & g(q) \leq 0. \end{aligned} \tag{2}$$

After collision and inverse kinematics (IK) validation is performed, the node is added to the graph's node list N as $n(x, q, c)$. While $x \in \mathbb{R}^3$ is the end-effector of the robot, $q \in \mathbb{R}^{3+m}$ denotes the optimized IK configuration. c equals to the cost value from the optimization solution. The sampling process stops when the number of N reaches a predefined number N_{node}.

2) Edge connections: As mentioned, we relax edge connection by assuming all edges are collision-free. For each sampled node in the list N, we connect to k nearest neighbours without checking for collision. Each edge is added to the graph with the cost defined in Eq. 3. The cost includes a combination of weighted Cartesian and configuration Euclidean distances between two end nodes, along with nodes' costs. w_x, w_q and w_c are the correspondent weights for the Cartesian, configuration distance and IK cost.

$$c_e = w_x \cdot \|x_1 - x_2\| + w_q \cdot \|q_1 - q_2\| + w_c \cdot (c_1 + c_2) \tag{3}$$

B. Path querying

1) Solution library: The reachability graph runs as an underlying service, waiting for the LGP planner to query two ends of a path. The service leverages a library to store previously computed solutions to avoid replanning for the same path and save planning time. During symbolic search in the first LGP's layer, the planner might iterate through different logical sequences and query a path in opposite

directions. For instance, a first call may seek a path from A to B, and subsequently, a solution from B to A. This is beneficial for tasks at the same levels, where one does not have a precondition on another action's completion.
The structure of each solution is shown in (4), where *key* contains the start and end point of the path, *path* and *cost* come from the resulting solution with n being the number of nodes on the path. Noted that the final configuration in *path* q_{1n} matches x_2 Cartesian pose. Each *key* is unique across the library in sorted order, and is used to query the data bidirectionally. On each call, the service checks the library and returns the stored solution. If the requested *key* is new to the library, the system starts to query the graph for the path. The answer is then added to the library for future reference.

$$\begin{aligned} \text{solution} = \{&\text{key: } (x_1, x_2)\ , x_i \in \mathbb{R}^3 \\ &\text{path: } [q_{10}, q_{11}, ..., q_{1n}] \in \mathbb{R}^{n\times(3+m)} \\ &\text{cost: } h\} \end{aligned} \tag{4}$$

In this data structure, *cost* is returned if the pipeline is querying heuristic cost, which is the function **heuristicCost** in Algorithm 1. When the framework reaches the final layer, *path* serves as guidance for trajectory optimization, accessed through the function **getWaypoints**.

2) Path planning: The reachability graph planner receives two Cartesian positions for each path querying message *key* as starting point and ending point, naming x_k and x_{k+1}. Nodes corresponding to x_k and x_{k+1} are added to the graph upon solving (2). Collision is checked for connecting these two nodes to the existing graph. If either x_k or x_{k+1} is unable to connect to the graph, we start enhancing the node. Gaussian sampling is applied with in-loop decreased covariance to find adjacent nodes as waypoints to connect to the existing graph. During this process, collision checking on both nodes and edges is performed to ensure a meaningful enhancement.

We use Dijkstra for finding the shortest path in the graph. We verify the path by checking collisions on all connected edges. We linearly interpolate the trajectory in configuration space with a predefined number of steps L. An edge with a detected collision is removed from the graph and replanning is triggered. In the case that any node loses all connections due to edge removal, it will be enhanced with the same method as starting/ending node enhancement. The difference is that the inspecting node can be replaced with an effective sampled one, which has the lowest IK cost c and is able to connect to the graph. In this sense, we ensure that the graph is fully connected, meaning that no subset is disconnected from the rest of the graph. During planning, the system updates the graph with enhancement and edge removal, benefiting future planning queries.

The flowchart in Fig. 2 shows in detail the path planning algorithm along with the interactions between the reachability graph and the LGP framework. The heuristic cost h is used internally in the symbolic search level. Before entering into the last stage of LGP, we interpolate $[q_k, q_{k+1}]$ ($k \in [0, T-1]$) with corresponding $[q_{k0}, .., q_{kn}]$ as input guidance for full path optimization.

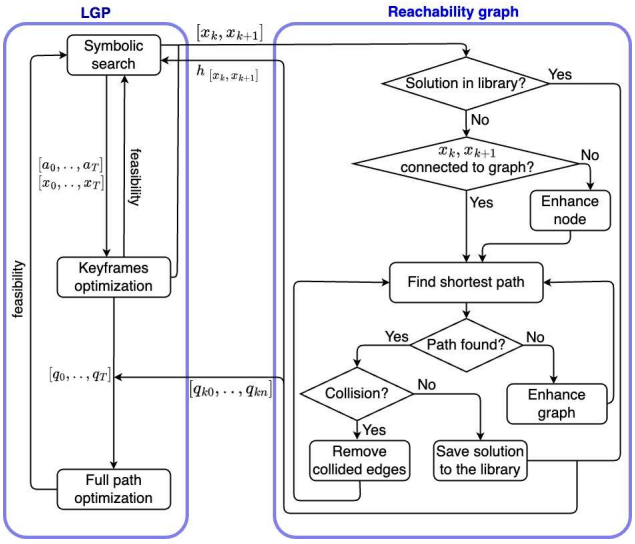


Fig. 2: Path querying system in R-LGP.

V. Evaluation

A. Simulation evaluation

In order to validate our approach, we propose a set of simulations and hardware experiments, performed on the PR2 and HSR robots. Our baseline for comparison is the heuristic-based LGP - RHH-LGP [16]. In addition, we compare our system against PDDLStream [2], the most popular and ready-to-use TAMP solver. We implemented the adaptive version of PDDLStream, which claims to outperform all other approaches for cost-sensitive problems. This allows us to set a priority on minimizing path length. There is an *optimal* flag in PDDLStream solver which determines whether or not the planner should explore more options and conclude with the best solution. We name *PDDLStream nonoptimal* and *PDDLStream optimal* for PDDLStream with *optimal* set to false and true respectively.

The experiments are designed to address the ability in accommodating geometric and kinematic constraints in integrated TAMP problems. We conducted an extensive evaluation with 100 runs in simulation for each experiment. All random tasks are intentionally feasible. The recorded metrics are:

- *Success rate*: the number of collision-free TAMP solutions over the total number of runs.
- *Planning time*: average time taken to get to successful solutions.
- *Path length*: average length, in meters, of the robot's base trajectory, assuming a floating-base robot.
- *Number of steps*: average number of steps to complete the given task. For this metric, we only consider key switches, e.g. pick, place, which intuitively correspond to the number of objects.

1) Pick and place: In this scenario, the PR2 robot is asked to pick and place 3 objects on the tables to the tray. This is a default TAMP task in the PDDLStream framework and is widely used in TAMP development [2]. The objects are at random positions on the table over 100 runs. We set a fixed

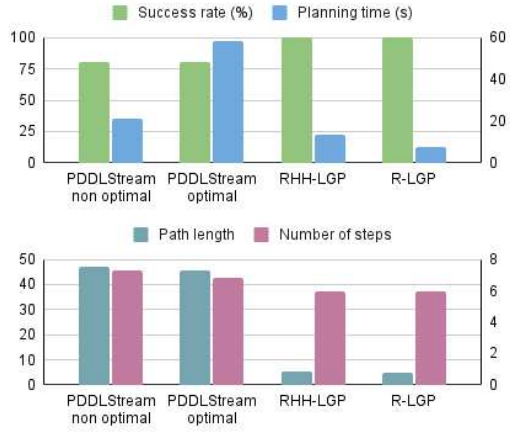


Fig. 3: Comparison result of the TAMP planners on pick and place task.

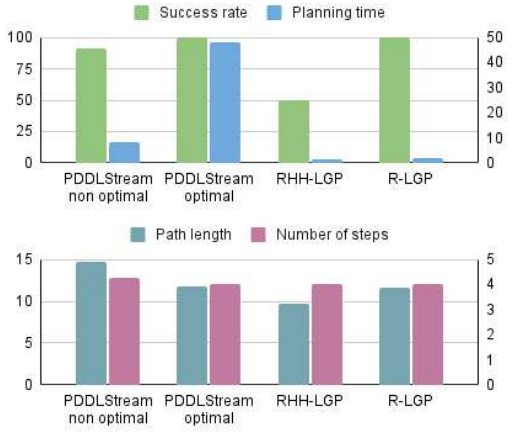


Fig. 4: Comparison result of the TAMP planners on sorting task.

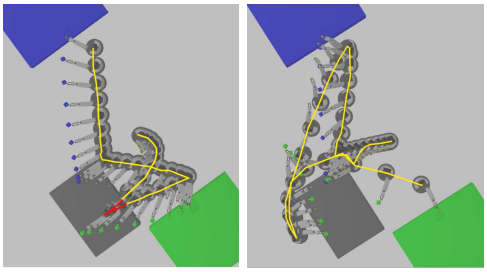
table size and all random poses are reachable from one side of the table.

The results in Fig. 3 show that the PDDLStream optimal solution is better in path length and step count than the default PDDLStream, although the latter achieves a shorter planning time. With PDDLStream solutions, PR2 tends to travel to the side of the table nearest to the object to pick it up regardless of reachability. LGP frameworks provide shorter trajectories by not navigating around the table, hence, also reduce the execution time. Apart from optimality, as opposed to PDDLSream, LGP frameworks achieve 100% success rate. Our R-LGP framework effectively reduces planning time leveraging the reachability heuristic.

2) Sorting: We increase the complexity of both task planning and motion planning in this task. The robot is asked to sort objects into trays with the same colors. Unlike the classic sorting task, we extend the experiment to address the reachability and mobility awareness of the system. In particular, we use larger table tops in this test, and randomized objects can be out-of-reach from one side of the table. This task requires TAMP solvers to accommodate obstacle-free configuration trajectories in solving high-level tasks, specifically requiring the robot to navigate around the table. To ensure all tasks are feasible, we constraint the objects' randomized locations to be within reachable distance from at least one side of the table.

In Fig. 4, it is worth noticing that the baseline RHH-LGP fails to plan a collision-free trajectory in 50% of the cases. Due to local optima in trajectory optimization, RHH-LGP planner can only solve tasks where the randomized location is within reach from the same side to the current position. This issue then filters runs with shorter path and results in a shortest average path length. PDDLStream optimal and R-LGP provide similar quality of final solutions with the same success rate of 100%. However, the planning time in R-LGP is significantly reduced from 48s to 2s.

Fig. 5 captures an example of TAMP results for the sorting task with the LGP baseline (a) and our planner (b). In the figures, the robot's base trajectory is shown in yellow line, which also denotes the resulted task sequence. In RHH-LGP,



(a) RHH-LGP (b) R-LGP

Fig. 5: TAMP results. Objects are initially spawned on the grey table and should be put on the table with the same color. R-LGP outperforms RHH-LGP in generating reachability-aware robot base trajectory (yellow lines) and avoiding collisions (red lines).

the planner considers the distance-based heuristic, hence, decides to sort the green block first. Meanwhile, our heuristic cost informs the high-level planner that the robot must go around the table in order to pick up the green block. Therefore, the decision to pick the blue block first produces a shorter travelling cost. With the help of the reachability graph, the symbolic planner can consider the cost of navigating taking into account reachability capabilities. In terms of full trajectory optimization, the red line shown in Fig. 5a is the violation of collision checking in RHH-LGP. We successfully solve this issue with the guidance from the reachability graph to the final layer in LGP, which enables the collision-free path in Fig. 5b.

B. Real robot validation

We validated our framework on a physical HSR. We implemented 2 tasks of increasing difficulty:

- *Sorting task*: the robot is asked to pick objects and place in the correct trays (Fig. 6).
- *Table clearing*: the task is to put objects from the surface into the drawer (Fig. 1).

In this real-world experiment, R-LGP proved to successfully solve a longer-horizon task with table

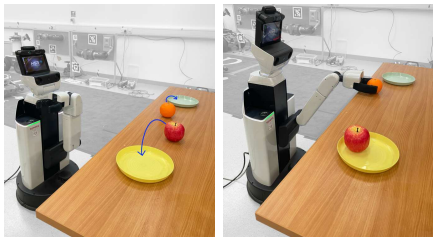


Fig. 6: Physical HSR using R-LGP solver in sorting task.

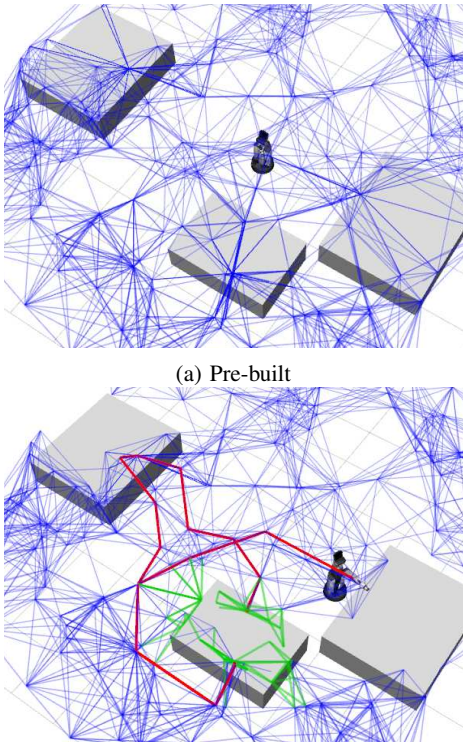
clearing. The desired symbolic sequence of this mission is (*take_knob, open_drawer, pick_object, drop_object, take_knob, close_drawer*). Figure 1 provides a visualization of the listed behaviours. Treating *take_knob* separate to *open_drawer* and *close_drawer*, the work can also be applied to opening/closing drawers, cabinets or doors. In our trial, the stair-like cabinet, as shown in Fig. 1, also poses a challenge of reachability awareness and obstacle avoidance. When the object is placed on the lower step, picking up action must consider that certain directions are blocked by the step above it.

In the sorting task, the R-LGP planner generated the result in 2s for 4 steps, with 2 objects correctly placed. It took longer for table clearing, 17s, to plan the long sequence with 6 separate steps, which the robot travelled 9m. In both examples, with our R-LGP solutions, the HSR robot can perform the task robustly with an optimal and collision-free trajectory.

C. Reachability graph evaluation

For each environment, we precompute the reachability graph only once. On average, the reachability graph is built within 14.2s with 200 samples over the specified workspaces.

We designed the sorting task to evaluate the robustness and completeness of our reachability graph, as this environment has more geometric constraints to consider during motion planning. Figure 7 visualizes our reachability graph for this scenario. The pre-built reachability graph is shown in Fig. 7a. In this step, nodes are validated with a collision-free configuration state and neighbour edges are all connected. The generated solution for the whole successful trajectory is plotted in the red line in Fig. 7b. The solution sufficiently produces a collision-free guidance path that has an understanding of the robot kinematics and reachability. This resulting path provides LGP with meaningful heuristic cost and trajectory guidance. The enhanced graph after a few queries is also denoted in Fig. 7b. The green lines represent added nodes and edges while searching for paths. In this example, the enhancement is mainly around the middle table, as objects for sorting are randomized on this table. The graph is updated over time, where edges in collision are removed and new nodes are added. This reduces planning time for subsequent queries in the same workspace.



(a) Pre-built

(b) After planning

Fig. 7: Reachability graph before and after planning. The red line is an example of the generated guided path, and green lines present the graph enhancement over time.

VI. Conclusion

R-LGP, our proposed framework, provides a robust solution to TAMP on mobile manipulators, which is tightly integrated to LGP. The introduction of the reachability graph not only successfully informs the LGP symbolic planner with information regarding reachability and mobility capabilities but also resolves local optima in trajectory optimization. The reachability-aware heuristic cost helps to prune out infeasible trajectories at the first layer, hence, reducing failures at the final full trajectory optimization. Despite being a sampling-based graph, it boosts the efficiency of both symbolic search and trajectory optimization without loss of optimality for LGP. As a result, the system is time-efficient in generating optimal TAMP solution while respecting robot geometry and kinematics. We presented an extensive evaluation to validate the framework, including real-world experiments on the Toyota HSR robot. In comparison to the state-of-the-art PDDLStream in the basic pick and place task, we reduced path length by 10 times with 6 times quicker solving time. The sorting task experiment highlighted the drawbacks of the current LGP framework, where environmental constraints can adversely affect symbolic planning. Without guidance, the baseline also fails to generate a collision-free trajectory. Hence, we achieved twice better success rate and improved completeness of the LGP solver. While sampling-based motion planning does not provide optimal solutions in limited time, our proposed reachability graph integrated to LGP can solve optimal TAMP. An avenue for future work is pruning task-specific constraints for TAMP on long-horizon applications. In our current framework, we are addressing the motion rather than task knowledge, and we believe pruning will help with scaling up to larger state spaces.

REFERENCES

[1] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[3] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[4] N. J. Nilsson, "Shakey the robot," *Technical note 323*, 1984.

[5] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.

[6] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 639–646.

[7] N. Dantam, S. Chaudhuri, and L. Kavraki, "The task motion kit," *IEEE Robotics & Automation Magazine*, vol. PP, pp. 1–1, 05 2018.

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach." in *Robotics: Science and systems*, vol. 12, 2016, p. 00052.

[9] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[11] K. Kim, D. Park, and M. J. Kim, "A reachability tree-based algorithm for robot task and motion planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3750–3756.

[12] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[13] K. T. Ly, M. Munks, W. Merkt, and I. Havoutis, "Asymptotically optimized multi-surface coverage path planning for loco-manipulation in inspection and monitoring," in *IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023.

[14] H. X. Li and B. C. Williams, "Generative planning for hybrid systems based on flow tubes," in *ICAPS*, 2008, pp. 206–213.

[15] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6886–6893.

[16] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "Rhh-lgp: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 761–13 768.

[17] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," in *RSS Workshop on Robust Task and Motion Planning*, 2019.

[18] A. T. Le, P. Kratzer, S. Hagenmayer, M. Toussaint, and J. Mainprice, "Hierarchical human-motion prediction and logic-geometric programming for minimal interference human-robot tasks," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 7–14.

[19] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[20] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8705–8711.

[21] K. E. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-integer and constraint programming techniques for mobile robot task planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 500–507, 2016.

[22] M. Conforti, G. Cornuéjols, G. Zambelli *et al.*, *Integer programming*. Springer, 2014, vol. 271.

[23] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annual Reviews in Control*, vol. 51, pp. 65–87, 2021.

[24] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[25] P. Culbertson, S. Bandyopadhyay, and M. Schwager, "Multi-robot assembly sequencing via discrete optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6502–6509.

[26] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1017–1023.

[27] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (tmit*): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 370–11 377, 2022.

[28] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, 2023.

[29] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Kim Tien Ly, Valeriy Semenov, Mattia Risiglione, Wolfgang Merkt, Ioannis Havoutis. "R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators". *Proc. IEEE International Conference on Robotics and Automation (ICRA),* 2024.

Student Confirmation

Student Name:	Kim Tien Ly		
Contribution to the Paper	• Developed the project idea • Wrote the code for the pipeline • Conducted the experiments • Collected data and performed analysis • Wrote large portions of the paper		
Signature		Date	04/10/2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Ioannis Havoutis			
Supervisor comments			
Signature		Date	07/10/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

Errata

I would like to note the following correction to the paper "R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators".

Location: Section IV.A.1., Equation 2.

Original text:

$$\begin{aligned} \min_{q} \; & M \cdot \max(0, g_{\text{prec}}(x_0)) + |x - x_0|^2 \\ \text{s.t.} \; & g_{\text{prec}}(x_0) \leq 0, \\ & x = f_{\text{kin}}(q), \\ & g(q) \leq 0. \end{aligned}$$

Corrected text:

$$\begin{aligned} \min_{q} \; & |x - x_0|^2 \\ \text{s.t.} \; & g_{\text{prec}}(x_0) \leq 0, \\ & x = f_{\text{kin}}(q), \\ & g(q) \leq 0. \end{aligned}$$

This correction does not affect the overall results or conclusions of the paper.

5.2 Discussion

In this paper, we proposed Reachability-guided Logic-Geometric Programming (R-LGP), an optimization-based reachability-aware TAMP framework for mobile manipulation. With a reachability graph developed as guidance to both the symbolic planner and trajectory optimization, our approach reduces the need for replanning, ensuring that the final trajectory optimization yields feasible and collision-free paths.

Our reachability graph is designed as a lazy-PRM structure that verifies joint feasibility with nodes being end-effector poses. The pre-generation phase involves checking whole-body feasibility in Cartesian and configuration space, which solves an IK formulation considering both geometric and kinematic constraints. For simplicity, edges are only checked during querying. The sampling-based graph captures the robot-specific requirements while ensuring sufficient exploration of the operating environment.

The results demonstrate the framework's effectiveness in both simulated and real-world environments. R-LGP consistently outperforms state-of-the-art TAMP solvers like PDDLStream [31], achieving higher success rates, reduced planning times, and shorter path lengths across mobile manipulation tasks. PDDLStream underperforms in path length and number of steps by outputting redundant actions (e.g. in our pick-and-place task, the robot picks an object and puts it somewhere instead of inside the tray), which essentially solves the problem from a feasibility perspective. Moreover, the success rate of PDDLStream is lower as it doesn't consider all object states during planning, and with a small tray, it fails to place all objects given the constrained space. Regarding LGP, the integration of the reachability graph efficiently boosts planning time and resolves local minima issues in the baseline, the heuristic-based LGP [75], hence increasing the success rate by twice in obstacle-rich environments. The framework was validated on the HSR in real-world experiments, where it completed complex tasks, such as object sorting and table clearing, with optimal, collision-free trajectories.

Our framework introduces a novel approach for integrating robot constraints into high-level planners, ensuring that these planners are informed of both the

robot's physical and kinematic limitations during task and motion planning. This approach leverages the strengths of both sampling-based and optimization-based techniques, combining the efficient exploration capabilities of sampling with the precision and optimality of optimization methods. The fusion of these techniques allows our framework to handle complex, high-dimensional problems, making it particularly well-suited for mobile manipulators to obtain robust and time-efficient TAMP solutions.

Despite the system's strengths, there are several limitations that should be considered. Firstly, although PRMs are probabilistically complete, this property does not hold in R-LGP because the reachability graph is constructed in the end-effector space rather than the configuration space. Another limitation we encountered during the development of our framework is the significant effort required to manually define tasks using a formal planning language, which involves specifying action predicates, effects, and the relationships between them. This process can be time-consuming with trials, as it requires precise descriptions of every possible action the robot can take and how these actions influence the environment. As a result, the manual task definition process introduces a potential bottleneck in scalability and flexibility. This limitation highlights the need for more automated or adaptive methods for task specification that could reduce the reliance on manually crafted formal descriptions, which motivates our next project in Chapter 6. Moreover, R-LGP remains an offline planner with a focus on optimal solutions. Developing an online and reactive planner is also our goal in Chapter 6.

6

Interactive Lightweight Robotics Planner

This chapter includes two papers that jointly target a generalizable TAMP with the following characteristics:

Task specification	Natural language
Interaction between task and motion planning layers	Decoupled
Optimality vs feasibility	Optimal
Open-loop vs closed-loop	Closed-loop
Role of human	Human-in-the-loop

Given the limitation of the TAMP solution in Chapter 5, where tasks and actions are manually defined by engineers with formal language, this chapter studies the use of LLM in the context of robotics planning. As LLM models are trained on large-scale web data, they obtain a good understanding of the world, which can be deployed into robotic systems in order for them to understand the tasks and actions' effects.

As the robot can obtain the world information from LLM, the next question is how this knowledge is personalized to the robot itself, enabling embodied intelligence. In this project, we study how robots can effectively make use of foundation models to obtain seamless human-robot interaction and how robot-centric constraints are informed in the system. The framework targets important characteristics of TAMP solutions including interpretability, reliability, robustness and scalability.

In this chapter, we introduce a multi-modal LLM-based robotics planner with a plug-and-play structure that facilitates the integration of SOTA modules.

- The model fuses human input with natural language task, perception, and feasibility modules (from the reachability graph designed in R-LGP - Chapter 5) to output an executable plan.
- We include a failure recovery pipeline with user instructions, taking advantage of human knowledge and perception. This study addresses online and interactive planning and makes a good combination of the plan-on-the-go feature of the work in Chapter 4 and the general-purpose domestic planner as in Chapter 5.
- The developed framework provides a light-weight LLM solution that allows real-time on-board computing.
- In addition, Section 6.2 includes the collaboration work with my substantial contribution that complements the main development of this chapter, which introduces a generalizable motion policy serving the motion library of the LLM-based planner.

6.1 InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy

. The article in this section was submitted for review. Our project website is available online at https://kimtienly.github.io/InteLiPlan

InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy

Kim Tien Ly[1], Kai Lu[2], Ioannis Havoutis[1]

***Abstract*— We introduce a lightweight LLM-based framework designed to enhance the autonomy and robustness of domestic robots, targeting onboard embodied intelligence. By addressing challenges such as kinematic constraints and dynamic environments, our approach reduces reliance on large-scale data and incorporates a robot-agnostic pipeline. Our framework, *InteLiPlan*, ensures that the LLM model's decision-making capabilities are effectively aligned with robotic functions, enhancing operational robustness and adaptability, while our human-in-the-loop mechanism allows for real-time human intervention in the case where the system fails. We evaluate our method in both simulation and on the real Toyota HSR robot. The results show that our method achieves a 93% success rate in the fetch me task completion with system failure recovery, outperforming the baseline method in a domestic environment. *InteLiPlan* achieves comparable performance to the state-of-the-art large-scale LLM-based robotics planner, while guaranteeing real-time onboard computing with embodied intelligence. More information about InteLiPlan can be found at the project website: https://kimtienly.github.io/InteLiPlan.**

I. Introduction

In recent years, the integration of artificial intelligence (AI) into robotics has led to significant advancements in automation and autonomous systems [1]–[5]. A key development in this field is the application of large language models (LLMs), such as GPT [6], [7], LLaMA [8], [9], and Mistral [10], which have proven to be powerful tools for understanding and generating human-like text. These models offer novel approaches for enhancing human-robot interaction and enabling more intuitive decision-making processes in robotics.

Despite their impressive capabilities, applying LLMs in robotics presents unique challenges [11]. These challenges stem primarily from the constraints of robot kinematics and the dynamic nature of the environments in which robots operate. Additionally, integrating these generative models within the existing robotics pipelines is non-trivial, as these are usually tightly coupled system of perception, planning, and actuation. Having such system running in real-time robotic deployment also poses a significant challenge due to the computational overhead. Moreover, robotic motion safety and robustness are critical concerns in human-robot environments. Therefore, developing a general-purpose autonomous robotic system in domestic scenarios remains a challenging open problem.

This paper proposes a lightweight LLM-based framework designed for domestic robots, aiming to address the aforementioned challenges. Our approach reduces reliance on large-scale data and develops a robot-agnostic pipeline. Furthermore, we detail strategies to mitigate kinematic constraints through real-time feasibility checks, ensuring that decision-making processes facilitated by LLM align effectively with the physical capabilities of robots.

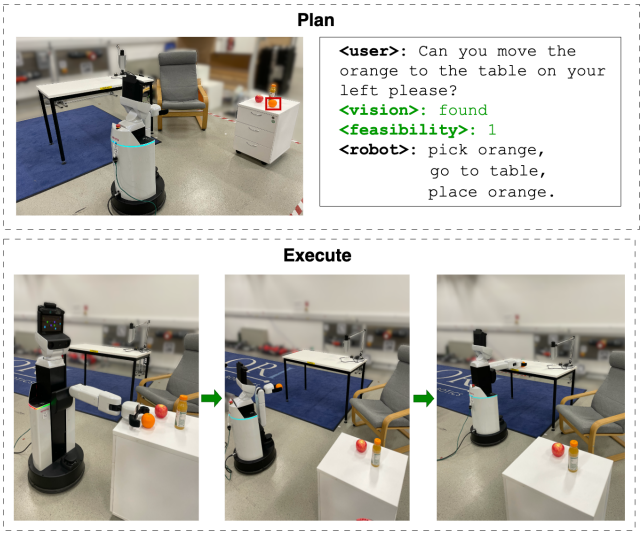


Fig. 1: Step-by-step execution of InteLiPlan result on the physical HSR. In our system, the robot receives requests and guidance from humans, e.g., pick-and-place of an orange. Our lightweight onboard planner will generate the robot actions for the task as shown in the figure.

An integral part of our framework is the human-in-the-loop (HITL) mechanism, which allows the system to handle system failures (e.g., partially observable environment, vision failures due to lightning, etc.) through human intervention. This feature is critical for maintaining autonomy, especially in complex environments where decisions by LLM may fail to execute. The system allows human operators to input corrective actions directly, facilitating on-the-fly adaptation to new or unanticipated situations. This real-time interaction not only enhances the robustness and reliability of robotic operations but also enables fine-tuning of the LLM's responses in accordance with human guidance.

By tackling these critical issues, our framework not only enhances the practical applicability of LLMs in robotics but also paves the way for more intuitive and responsive robotic systems capable of complex interactions and tasks in varied real-world scenarios. In summary, our contributions are: noitemsep

- An interactive robotics task planner that integrates user command, visual perception, and feasibility score mod-

[1]: K. T. Ly and I. Havoutis are with the Oxford Robotics Institute, University of Oxford. Email: {ktien,ioannis}@robots.ox.ac.uk.
[2]: K. Lu is with the Department of Computer Science, University of Oxford. Email: kai.lu@cs.ox.ac.uk.

ules into a cohesive multimodal framework.
- A human-in-the-loop planner that allows intuitive user intervention for robust failure recovery, enhancing the autonomy of the robotic system at hand.
- A lightweight solution to real-time onboard LLM-based robotics planner.

II. Related work

A. LLMs for Robotic Systems

The application of LLM models in robotics, driven by recent advancements in natural language processing, has become a focal point of intensive research [12]. Silver et al. [13] integrates a large language model (LLM) to interface with planning domain definition language (PDDL). Similarly, Plansformer [4] requires PDDL domain inputs and fine-tunes the transformer model on planning problems. Although the PDDL interface is a neat approach to use with the current task planner, it is difficult for end-users to add new tasks. Alternatively, Liang et al. [5] utilize LLM for generating code policies that can be executed on the robot. However, they do not account for the robot's kinematics or geometry in the planning process, which is critical in real-world applications.

Conversely, SayCan [1] and Text2Motion [14] are notable for considering geometric feasibility when planning the action sequence. PIGINET [15] instead introduces a transformer-based plan feasibility predictor to be integrated in a TAMP planner. The method fuses 6 cameras, making it harder to deploy in domestic setting. Recently, applying vision-language models (VLMs) [16]–[18] in robotics is another promising direction. Robotics transformer (RT)-2 [3] employs vision to enable a more generalizable approach to its former version RT-1 [2]. Although these models report robust embodied intelligence, their large size necessitates reliance on cloud-based systems. In contrast, we explore how onboard, LLM-based models can achieve similar cognitive capabilities while operating independently.

B. Human-Robot Interaction for Autonomy

Recent research indicates that LLMs may not fully replicate human reasoning capabilities [19]. Consequently, integrating HITL proves essential for maintaining the reliability and safety of robotic systems [20]. Vemprala et al. [7] implement ChatGPT with HITL, emphasizing the dual benefits of enhanced model training through human feedback and increased safety during operations. And Ren et al. [21] focus on human-robot interaction, where the robots decide whether or not to ask for human intervention. Meanwhile, our platform incorporates multimodal structures that consider human interaction, vision, and action feasibility.

C. Replanning for Failure Recovery

Failure recovery has been addressed widely in literature for robotics planners. When it comes to LLM-based solutions, researchers typically address replanning from a second-level action check when the plan is generated [22] [23]. Vision-language models are also implemented to describe the error from the scene [24]. On the contrary, we designed a multimodal LLM-based planner that incorporates action feasibility during generating the plans, which helps to reduce the time taken for re-querying the planner.

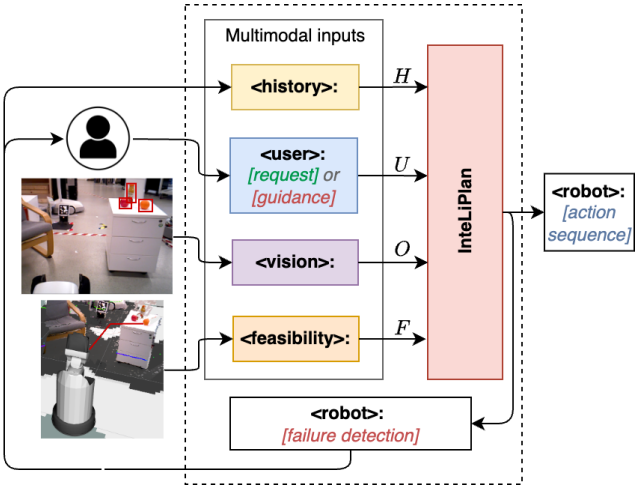


Fig. 2: System overview. Our multimodal planner integrates user command, visual perception, and action feasibility score as inputs to a fine-tuned lightweight LLM. The LLM will then generate an action sequence for a real robot to perform mobile manipulation tasks.

D. Data-efficiency in LLM-based robotics planner

LLM-based models are usually data-intensive, demanding substantial training datasets to perform optimally. In robotics, this demand poses a significant challenge, as collecting large, diverse, and representative datasets from physical robots is time-consuming, labor-intensive, and costly. For instance, RT series [2], [3] introduces an end-to-end approach to applying transformers to robotics, requiring a large amount of data collected in seventeen months to map language to actions. Similarly, Lynch et al. [25] proposes an interactive method for language-conditioned robot skills, compiling a dataset through 2.7k hours of collection on real robots. Prompt engineering, on the other hand, has shown promise in applying LLM to robotic tasks in zero-shot manner [26]–[28]. However, it is not robust in real-world execution without properly understanding the operating environment. Our work introduces a widely applicable LLM-based framework that is easy to deploy on different robotics platforms, as it does not require robot-specific data. Therefore, the framework guarantee reproducibility.

III. Methodology

A. Problem Statement

Our work involves a mobile manipulator interacting with non-expert users to perform a human request. During operation, the robot receives input from the user U, which can be either a request or guidance for the robotic agent. The centralized planner, *InteLiPlan*, takes in the user input and the internal vision and motion verification functions, in order to generate a feasible plan. If a plan is found, the sequence of actions will be executed from a predefined skill library. If a failure is detected, the robot will notify the reason and

ask for human instruction. The skill library defines a list of one-step actions such as picking or placing objects.

The robot obtains environmental information from a vision module, which is a state-of-the-art object detection model, processing from the onboard RGB-D camera. The visual detection is represented as O. The motion verification module assesses the feasibility score F of the action during planning.

Our work studies a real-time onboard LLM-based robotics planner, with the ability to interact with human in natural language and re-planning upon guidance. InteLiPlan is evaluated with domestic tasks, aiming at a human-centric and interpretable framework.

B. Interactive Planner with Multimodal Perception

Our planner utilizes a fine-tuned LLM model, which takes multimodal inputs from the user (U), visual observation (O), and motion feasibility (F), as shown in Fig. 2. Before feeding to the central planner, the system evaluates the inputs in the following order: U, O, F. Algorithm 1 explains the internal interaction between the modules. When the user sends a command to the central planner, the object list is extracted and sent to the vision module. Then, the object detector runs as a service that checks if the mentioned items are found in the current observation. The motion verification module depends on the vision module to get the object position, which acts as the desired end-effector goal to execute the action. We keep track of the conversation by inserting a history H to each planner call. H is initialized empty and records the human-robot interaction over time.

Algorithm 1 InteLiPlan pipeline

Require: user U, robot R, interaction history H, vision module O, feasibility module F

inputs $\leftarrow U$.input()
objects_list $\leftarrow R$.extract(*inputs*)
$H \leftarrow$ None
while not (R.task_is_complete() or time_out()) **do**
    objects_loc $\leftarrow O$.get_position(*objects_list)*
    result $\leftarrow R$.plan(H, *inputs*, O.is_seen(*objects_list*), F.get_score(*objects_loc*))
    if R.get_failure(*result*) **then**
        $H \leftarrow H$ + *inputs* + R.get_failure(*result*)
        inputs $\leftarrow U$.get_guidance()
    else
        R.execute(R.get_plan(*result*))

InteLiPlan is formulated as $A = \phi(H, U, O, F)$, where $A = \{a_t, ..., a_{t+T}\}$ representing the outputted sequence of actions. While U is the direct input from the user, H, O and F are the internal conversations of the system. H is a quoted string that tracks the previous conversation. We design O as a text-based feedback from the observation, and $F \in \{0, 1\}$ as a binary signal indicating the feasibility of the action. The multimodal inputs to the planner are represented in specific tokens: <*history*>, <*user*>, <*vision*>, <*feasibility*>, where the token for the output is <*robot*>. One example of the multimodal conversation is shown in Fig. 1.

Our framework features the integration and coordination between submodules, which are then utilized in an LLM-based planner. We provide a plug-and-play modular design to leverage state-of-the-art (SOTA) models across various research domains in a zero-shot way. The design helps to reduce reliant on robot-specific data, hence, improves adaptability to new system.

C. Failure Recovery

For real-world robotic applications, the ability to replan tasks and motion upon a failure is a critical feature. This can be accomplished by active exploration, or with guidance from humans. A recent paper from [29] solves a self-recovery problem with an LLM-based planner. Our work instead addresses cases where system failures occur. Hence, we blend our method with the human-in-the-loop mechanism, taking advantage of human input. When a failure occurs, whether due to vision limitations or kinematic constraints, our system can intelligently reassess and modify its output action sequence based on human guidance. This adaptive policy not only helps overcome immediate obstacles but also refines the robot's decision-making processes over time, enhancing the efficiency and safety of robotic operations in domestic environments.

The multimodal structure allows the planner to reason around failures through feedback from the vision and motion verification modules. These modules can also provide contextual failure descriptions to the user, enhancing the system's explainability and enabling informed human intervention. For instance, vision failures due to partial observation or adverse lighting conditions can be mitigated with human insights, leveraging the planner's capability to maintain a record of initial commands through the internal H and to replan upon system failure as necessary, as demonstrated in Algorithm 1.

D. Text-only Fine-Tuning Data for Robotics Planner

Recently, prompt engineering has become popular for implementing ready-to-use LLM-based planners, but it typically requires extensive trials or multiple human-agent interactions to clarify tasks. Besides, outputs from such planners often need translation into robotic control commands. Therefore, we choose to fine-tune an LLM for our multimodal robotic planner that can generate the desired format for the real robot's action API calls. As *InteLiPlan* is designed for onboard computing, this also reduces the need for an extra conversion module which takes up computing resources.

To fine-tune the LLM model, we collected a pure-text customized dataset that includes scenarios both with and without failure recovery, where the interactions in failure cases are limited to a maximum of two rounds. The dataset is formulated as $D = \{H_i, U_i, O_i, F_i, A_i | i = 0, 1, 2, ...N\}$. We loop through a list of objects and commands to generate the command and expected sequence of action. During fine-tuning, visual observation O_i and feasibility score F_i is provided in the form of text. This text-based-only fine-tuning approach enables the model to easily adapt to different robots without further modifications or re-parameterization.

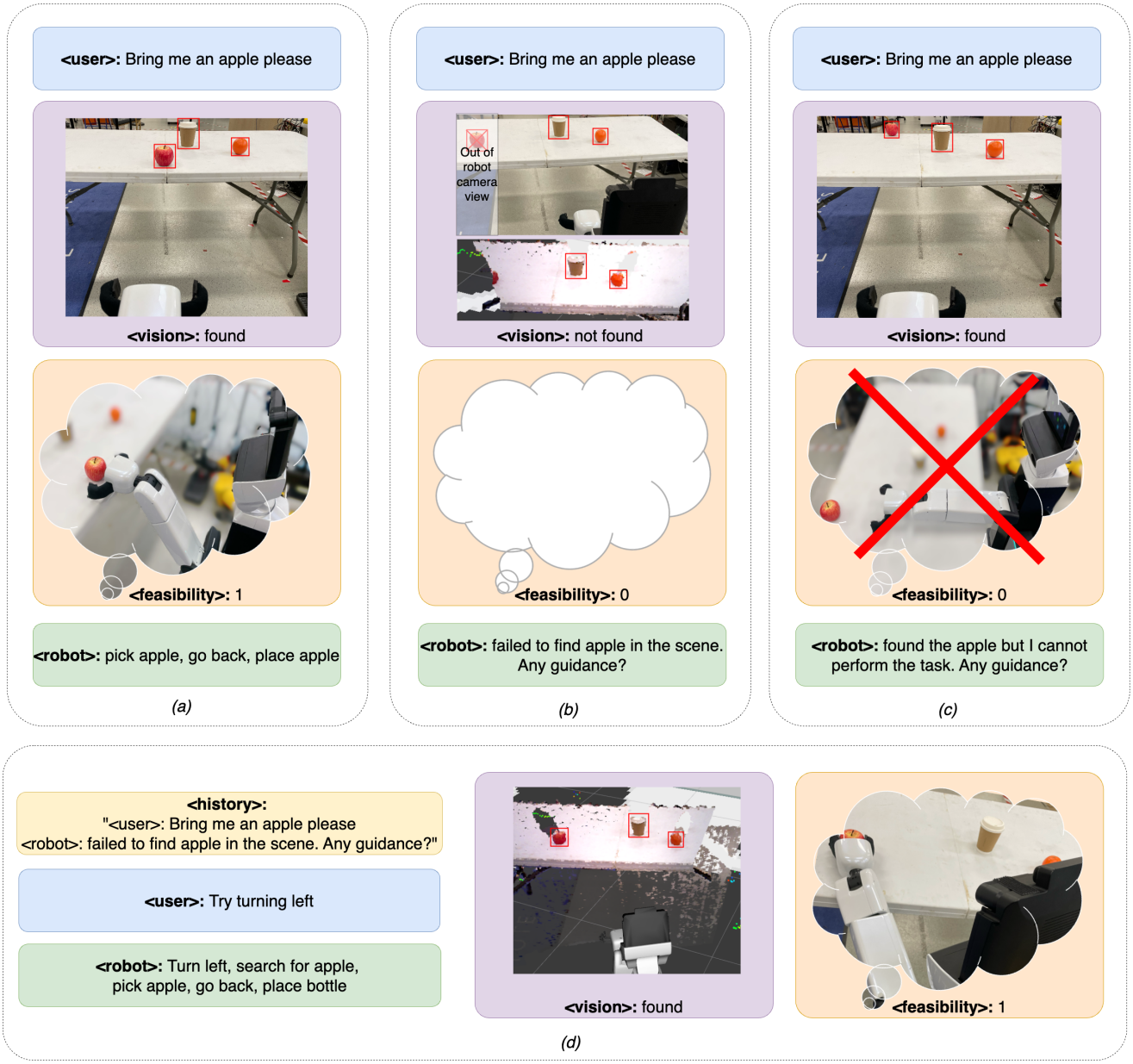


Fig. 3: Examples of the multimodal LLM-based planner. (a) presents a no-failure case for the given command. (b) and (c) depict the failure reasoning ability of the model considering the inputs, for vision and feasibility failures respectively. (d) showcases the ability to recover from the failure in (b) with human instruction.

IV. Experimental Results

Experimental Setup. We evaluate our approach using the Toyota Human Support Robot (HSR) in a domestic environment. The first set of experiment is a 'fetch me' task, which environment includes 20 seen objects and 20 unseen objects, and the planner will be evaluated on 1 seen command and 15 unseen commands. The unseen commands represent the variants in human-like conversation. We then scale up the model to test with the 101 task requests from SayCan [1]. The vision module contains an object recognition model implemented with YOLO [30] and a pose estimation process, which received inputs from the RGB-D camera equipped on the HSR. In this way, the robot automatically verified the presence and precise location of objects relative to the robot's map frame. For feasibility verification, we utilized a reachability graph from R-LGP [31], a sampling-based approach that efficiently checks for path from a starting point to a goal while ensuring feasible configurations for the robot reaching towards the target object. The LLM model for our approach is fine-tuned from Mistral 7B [10], a state-of-the-art lightweight LLM model, which is suitable for onboard deployment.

Metrics. To assess the performance of the proposed method we measure three metrics. 1) *Plan success rate*: the ratio of the trials where the robot successfully generates the plan in text. 2) *Execute success rate*: the ratio of the

trials where the robot successfully plans and executes the given request. 3) *Inference time*: the time taken to proceed the formulated multimodal inputs, showcasing the real-time capability of the model.

Baselines. We compare the work with few-shot prompting on Mistral 7B, *Mistral-Prompt*. The samples are in the same form as the dataset for fine-tuning, as discussed in Section III-D. The second comparison is with *SayCan* [1]. Given that this method does not have failure recovery ability, we only report the result during the scalability experiments. Generative result from *PaLM 8B* [32] is also reported as a lightweight LLM baseline.

A. Comparison of Different Methods in Feasible Tasks

In our first experiment, we assume successful outcomes in both object detection and robotic motion feasibility. Here, the planner is solely required to generate the appropriate action sequences based on human inputs. An example of such conversation is provided in Fig. 3a.

Despite being told to list a sequence of actions only, *Mistral-Prompt* outputs include redundant text, which is ignored when determining the success rate. As presented in the *w/o failure* column of Table I, the results demonstrate a superior performance of our model, achieving 100% success in scenarios with seen commands and objects, and maintaining high effectiveness even with unseen objects or commands. In contrast, *Mistral-Prompt* experienced performance drops in scenarios involving unseen elements, with a more sensitivity over unseen textual commands.

B. Evaluation of Robustness in Failure Recovery

In this section, we evaluate the ability of our framework's capacity to effectively replan from a detected failure, informed by the vision and feasibility modules.

The *w/ failure* column of Table I presents the successful task completion rate of the methods. The failure cases can come from either false failure detection or false replan strategy. The comparison shows a significant difference between *Mistral-Prompt* and ours. Fine-tuning Mistral overall creates a better adaptation to the task than few-shot prompting. While our method gained 93% success rate with unseen commands and unseen objects, *Mistral-Prompt* witnessed a significant drop, with 55% task success rate, in failure recovery experiments.

We further evaluate the breakdown of failure recovery components by reporting both the success rate of accurately identifying the cause of failure (*Failure detection*, as in Fig. 3b,c) and the success rate of task replanning based on human instructions (*Replan*, , as in Fig. 3d). This test considers the success rate based on the ability of the model to answer the correct situation. The results, as summarized in Fig. 4, demonstrate our framework's performance in both detecting failures and replanning in various scenarios. Notably, our method achieved perfect detection and replanning success rates in scenarios with seen commands and seen/unseen objects, underscoring its reliability in familiar settings. Even in more complex situations involving unseen commands with seen/unseen objects, our framework maintained a 100% success rate in failure detection and a 93% success rate in replanning. This contrasts sharply with *Mistral-Prompt*, which has 74% success rate in failure detection and replan. These results indicate the robustness of our system in real-world conditions where adaptability and responsiveness to failures are crucial for operational success.

TABLE I: Success Rates (%) of Task Completion.

Scenario	Method	w/o failure	w/ failure
Seen cmd + Seen obj	Mistral-Prompt	100	100
	Ours	100	100
Seen cmd + Unseen obj	Mistral-Prompt	95	66
	Ours	100	100
Unseen cmd + Seen obj	Mistral-Prompt	73	69
	Ours	100	93
Unseen cmd + Unseen obj	Mistral-Prompt	70	55
	Ours	100	93

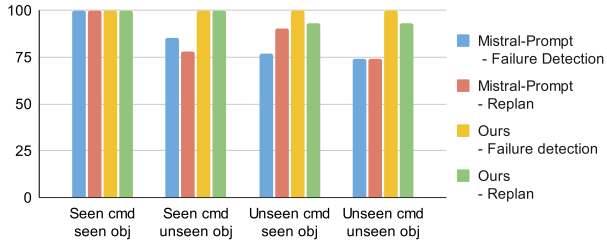


Fig. 4: Comparison of failure detection and replanning success rate between the methods across different cases.

C. Evaluation of Scalability

In this experiment, we evaluate the scalability of the approach by 101 task instructions from SayCan [1]. To accommodate the task variants, we fine-tuned Mistral with the tasks outlined in Table II, which covers the expected actions required for the dataset solutions.

TABLE II: Test Cases with Corresponding Expected Outcome.

Case	Expected plan
Pick object	Pick object
Go to destination	Go to destination
Fetch me	Pick object, Go back, Place object
Put away	Pick object, Go to destination, Place object
Put in drawer	Open drawer, Pick object, Place in drawer, Close drawer

Table III presents the tested result of the frameworks with the dataset from *Saycan*. As a 540B model, the *SayCan* result explains its excellent decision-making capability with throughout understanding of the world. Its trained affordance value provides sufficient embodied knowledge, ensuring high success rate in execution. Besides, we note that prompting with *PaLM 8B* only successfully plans 38% cases. *Mistral-Prompt* with our modular structure helps the LLM model to gain embodied intelligence, with the planning success rate to 59%. It is observed that *Mistral-Prompt*'s failure cases come from its lack of sense of the operating environment.

For example, some of the results tell the robot to 'go to store' to pick something up, despite being told that it is working in a domestic environment. It is proven that with 7B parameters, our *InteLiPlan* obtains similar results in comparison to the state-of-the-art 504B SayCan model with 83% planning success rate.

The execution success rate remains relative to the planning success rate in *Mistral-Prompt* and our framework, given the same vision and reachability modules check for the system. Having the reachability graph checking whole-body feasibility explicitly instead of using probabilistic, our approach also increases the chances of successfully executing the plan once generated. This is shown by the reduced difference in plan and execute success rates between *SayCan* and our pipeline. On the other hand, as we use a sampling-based approach, there is a risk of not finding a path even if there is one exists, leading to failures from the planning level. This is where the failure recovery comes in. Another straightforward solution is to use more samples in the graph such that they sufficiently provide the reachability knowledge. Since InteLiPlan is a plug-and-play system, it can seamlessly integrate future state-of-the-art feasibility modules.

TABLE III: Plan and Execution Success Rates (%) of the methods for 101 SayCan task descriptions.

Methods	Plan	Execute
PaLM 8B	38	n/a
PaLm-SayCan 504B	84	74
Mistral-Prompt 7B	59	58
Ours	83	82

D. Evaluation in Real-time Onboard Computing

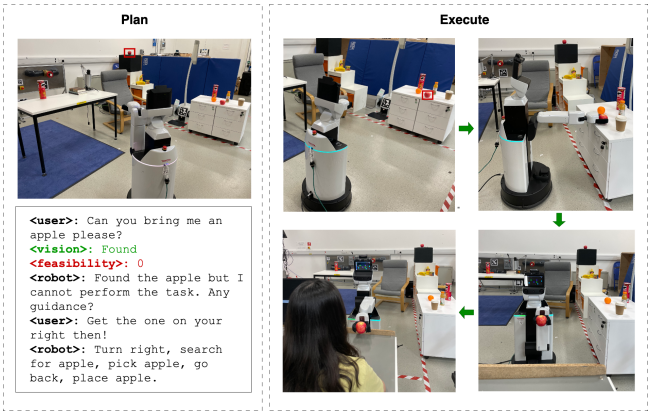
Our framework targets a lightweight transformer model to embed on a mini-computer, allowing robots to operate independently of networks. Hence, we validate the efficiency of the system on the physical HSR. Figure 1 and Fig. 5 showcase InteLiPlan results in real-world settings, which includes tasks with and without failure recovery. Videos of our demonstrations are available at our project page: https://kimtienly.github.io/InteLiPlan.

Table IV records the averaged planning time of the system over the trials, which breakdowns into vision, feasibility and planner queries processing time. In total, the model can plan an executable action sequence in less than 7s, guaranteeing real-time onboard operation.

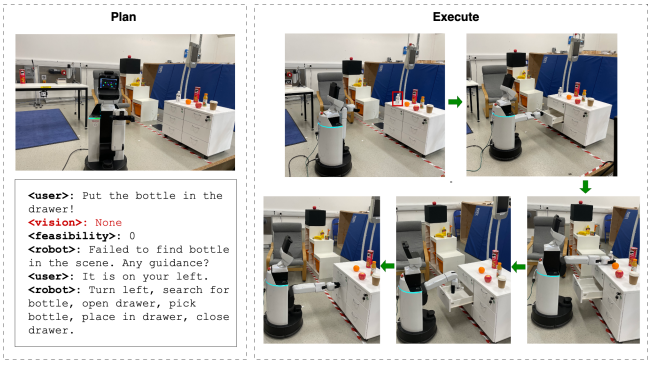
V. Conclusion

We presented *InteLiPlan*, an interactive lightweight LLM-based robotics planner for reliable and robust autonomy in domestic environments. Our framework employs a conversation-like format between internal modules and facilitates human-robot interaction, allowing the system to reason about failures through a multimodal input formulation. By incorporating a human in the loop, the robot gains the ability to replan based on instructions, effectively utilizing human input for dynamic adjustments.

With fewer than 200 data samples for fine-tuning, our approach effectively handles the targeted tasks by leveraging



(a) Failure from feasibility



(b) Failure from vision

Fig. 5: Demonstration of the interactive failure recovery capability on the physical HSR.

TABLE IV: Breakdown of onboard planning time (s).

Vision query	Feasibility query	Planner query
6e-6	5	1.5

the human-like text understanding capabilities of LLMs. This reduces the effort of defining traditional task planning domains in robotics, and provides an intuitive, robot-independent data structure. Impressively, deploying our approach with the lightweight Mistral 7B model achieves both comparable results with the SOTA baseline and – notably to our approach – real-time onboard computing. Our system was validated on the physical Toyota HSR robot.

Limitations and Future Work - Our framework is structured as a multimodal system with a feasibility check and can react to movable obstacles by asking humans for input. However, it does not have a motion-level re-planning feature. Incorporating low-level reactions together with high-level re-planning capabilities in the form of a dual process can potentially improve failure recovery speed and reduce human intervention. Furthermore, the sampled path formed by the reachability graph can be further developed into a motion planning subpart of our approach, resulting into a full task and motion planning stack.

REFERENCES

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[2] 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.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[4] V. Pallagani, B. Muppasani, K. Murugesan, F. Rossi, L. Horesh, B. Srivastava, F. Fabiano, and A. Loreggia, "Plansformer: Generating symbolic plans using transformers," *arXiv preprint arXiv:2212.08681*, 2022.

[5] 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)*. IEEE, 2023, pp. 9493–9500.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[7] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," Microsoft, Tech. Rep. MSR-TR-2023-8, February 2023.

[8] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[9] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[10] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[11] J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang *et al.*, "Large language models for robotics: Opportunities, challenges, and perspectives," *arXiv preprint arXiv:2401.04334*, 2024.

[12] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," *arXiv preprint arXiv:2311.07226*, 2023.

[13] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "Pddl planning with pretrained large language models," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[14] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[15] Z. Yang, C. R. Garrett, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," *arXiv preprint arXiv:2211.01576*, 2022.

[16] 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*. PMLR, 2021, pp. 8748–8763.

[17] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*. PMLR, 2022, pp. 12 888–12 900.

[18] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on robot learning*. PMLR, 2022, pp. 894–906.

[19] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," *arXiv preprint arXiv:2307.10169*, 2023.

[20] C. Zhang, J. Chen, J. Li, Y. Peng, and Z. Mao, "Large language models for human-robot interaction: A review," *Biomimetic Intelligence and Robotics*, p. 100131, 2023.

[21] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *arXiv preprint arXiv:2307.01928*, 2023.

[22] M. Ahn, M. G. Arenas, M. Bennice, N. Brown, C. Chan, B. David, A. Francis, G. Gonzalez, R. Hessmer, T. Jackson *et al.*, "Vader: Visual affordance detection and error recovery for multi robot human collaboration," *arXiv preprint arXiv:2405.16021*, 2024.

[23] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "Copal: corrective planning of robot actions with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8664–8670.

[24] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," *arXiv preprint arXiv:2302.01560*, 2023.

[25] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters*, 2023.

[26] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[27] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[28] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[29] M. Shirasaka, T. Matsushima, S. Tsunashima, Y. Ikeda, A. Horo, S. Ikoma, C. Tsuji, H. Wada, T. Omija, D. Komukai *et al.*, "Self-recovery prompting: Promptable general purpose service robot system with foundation models and self-recovery," *arXiv preprint arXiv:2309.14425*, 2023.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[31] K. T. Ly, V. Semenov, M. Risiglione, W. Merkt, and I. Havoutis, "R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators," 2024.

[32] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

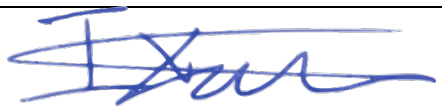
Title of Paper	InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy
Publication Status	□ Published □ Accepted for Publication x Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Kim Tien Ly, Kai Lu, Ioannis Havoutis. "InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy". *Under Review*.

Student Confirmation

Student Name:	Kim Tien Ly		
Contribution to the Paper	• Developed the project idea • Wrote the code for the pipeline • Conducted the experiments • Collected data and performed analysis • Wrote large portions of the paper		
Signature		Date	04/10/2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Ioannis Havoutis			
Supervisor comments			
Signature		Date	07/10/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

Addendum

I would like to add further explanation to the paper "InteLiPlan: Interactive Lightweight LLM-Based Planner for Domestic Robot Autonomy".

- **How is feasibility score computed?**

  We are using the reachability graph designed in R-LGP (Chapter 5 to get the feasibility score. Instead of verifying the full action sequence, the feasibility module takes in the current position and the end-effector goal position (e.g. object pose from the vision module), then checks if there is a collision-free trajectory to reach the target. If the object is not in the object detection list, the feasibility module returns 0 for the feasibility score.

- **How is LLM output parsed to an executable command to the robot or a question to the user?**

  We use keywords to detect if the model returns a failure or successfully plans an action sequence. By fine-tuning, the model learns to report failures or generate a sequence of actions in the format provided in the data. If a plan is found, the outputted sequence is broken down into steps by commas, and the system calls the pre-defined motion APIs. In the case that the model outputs a non-existent action, the robot will fail by not being able to access the relevant action function. If a failure is detected, the output will be directed to the user, explaining the reason for failure.

6.2 Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning

In this section, I discuss the collaborative development of a motion planner designed to complement InteLiPlan. InteLiPlan's reliance on exclusively text-based fine-tuning data allows its deployment on diverse robotic platforms. Integrating a generalizable motion policy as the low-level controller complements this adaptability by dynamically adjusting configuration trajectory to specified conditions. This integration supports the idea of a plug-and-play TAMP framework, promoting efficient execution across different robotics setups.

The paper employs adapters in RL to fine-tune pre-trained manipulation policies, enabling generalization across different tasks (e.g., drawer opening vs. door opening) and robots. This can complement InteLiPlan by providing a robust, low-level motion policy layer that adapts to each new platform's kinematic constraints, enhancing InteLiPlan's modular capability.

A core advantage of this approach is the low-level motion policy can be optimized to interpret and carry out high-level instructions while accounting for physical differences. This generalizability not only minimizes integration overhead but also enhances the system's ability to operate in diverse and unstructured environments, making it especially suitable for real-world deployment. Consequently, this easy-to-deploy motion library solution bridges the gap between abstract planning and adaptable actions in InteLiPlan.

The article in this section was presented at the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2024 [137]. Our project website is available online at https://kl-research.github.io/genrob

Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning

Kai Lu[1], Kim Tien Ly[2], William Hebberd[2], Kaichen Zhou[1], Ioannis Havoutis[2], Andrew Markham[1]

***Abstract*— This study investigates the use of adapters in reinforcement learning for robotic skill generalization across multiple robots and tasks. Traditional methods are typically reliant on robot-specific retraining and face challenges such as efficiency and adaptability, particularly when scaling to robots with varying kinematics. We propose an alternative approach where a disembodied (virtual) hand manipulator learns a task (i.e., an abstract skill) and then transfers it to various robots with different kinematic constraints without retraining the entire model (i.e., the concrete, physical implementation of the skill). Whilst adapters are commonly used in other domains with strong supervision available, we show how weaker feedback from robotic control can be used to optimize task execution by preserving the abstract skill dynamics whilst adapting to new robotic domains. We demonstrate the effectiveness of our method with experiments conducted in the SAPIEN ManiSkill environment, showing improvements in generalization and task success rates. All code, data, and additional videos are at this GitHub link: https://kl-research.github.io/genrob.**

I. Introduction

Learning generalizable robotic skills is a significant challenge in embodied intelligence, which includes generalization across objects, tasks, and robots. Compared to the widely explored vision-based object generalization [1], generalization across different robots and different task trajectories remains underexplored. This capability has a wide impact, enabling robots to efficiently learn new skills or adapt existing skills to similar domains. However, different robots usually have varying kinematic configurations and morphologies, such as body structure and joint limits, leading to different physical constraints and dynamic properties, posing challenges to skill generalization.

Traditional skill learning methods often consider retraining for new tasks on a specific single robot, such as by using reinforcement learning (RL) [2]–[4] or imitation learning [5]. However, reinforcement learning often encounters problems with sampling efficiency, and imitation learning struggles to find perfectly corresponding robot samples. Recent works designed skill generalization methods across multiple robots and tasks using large robot learning datasets such as Open X-embodiment [6] and foundation models [7], [8], which requires significant computational resources. Another approach is the use of hierarchical or modular network designs [9]–[13], including aligning internal features [9], encoding robotic morphological information [10], and sharing modular policies [11], [12]. However, they often focus on simplified robotic morphologies, such as 2D arms [11] or omnidirectional spherical hands [13], or limited tasks like grasping [10]. In this paper, we aim to explore how a shared global skill policy can effectively be applied on multiple high-degree-of-freedom (high-DoF) mobile robot platforms.

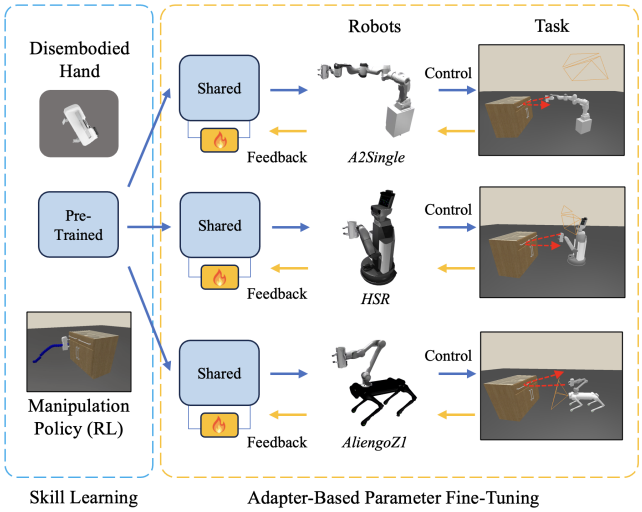


Fig. 1. **Demonstration of the proposed method.** In this work, we study the problem of using adapter-based fine-tuning on pre-trained policy models for generalizable manipulation across different robotic platforms. We teach a disembodied hand to learn tasks like opening a drawer and transfer these skills to a whole-body robot, accounting for the robot's specific constraints.

Our key innovation is to teach an unconstrained, disembodied hand manipulator to learn a skill, such as opening a drawer or cabinet, and then transfer this skill to a constrained whole-body robot. We consider these constraints as the feasibility of the disembodied hand's trajectory on a specific robot. As the disembodied hand policy is not optimized for the specific robot's constraints and kinematic properties, the trajectories generated by this policy are not optimal, or even unfeasible, on new whole-body robots. For instance, a robot with more DoFs and longer arms will have greater adaptability, while smaller robots may find it more difficult to follow the poses generated online by RL. To mitigate this issue, we consider fine-tuning the existing policy network and introducing robotic control feedback to optimize the RL policy.

Recently, parameter-efficient fine-tuning (PEFT), such as the Adapter technique [14]–[17], has proven effective for pre-trained model fine-tuning and is widely used for generalization in natural language processing [18] and visual generation fields [14]. It achieves specific domain adaptation by inserting additional trainable layers into the existing model. This method allows the model to adapt to new domains with a small number of extra parameters while keeping

[1]: K. Lu, K. Zhou and A. Markham are with the Department of Computer Science, University of Oxford. Email: {kai.lu, rui.zhou,andrew.markham}@cs.ox.ac.uk

[2]: K. T. Ly, W. Hebberd and I. Havoutis are with the Oxford Robotics Institute, University of Oxford, Oxford, UK. Email: {ktien, william.hebberd,ioannis}@robots.ox.ac.uk

the original parameters unchanged, which is particularly useful in transfer learning and multi-task learning. Inspired by this, we propose the integration of adapters into the learning of generalizable robotic skills and explore the use of robot feedback in RL to learn these adapters. This approach enables the adaptation to new robots or tasks without the necessity of retraining the entire model. We conjecture that this method can better retain the original model's knowledge of skill dynamics while introducing an understanding of new robots or tasks.

To implement this, we introduce parallel modules into the original network, such as low-rank decomposed (LoRA) [19] adapters, or sequential modules, such as low-dimensional encoder-decoder structure adapters with residual design [20]. Then, in RL training, we design a feedback reward function from the whole-body robotic control, which requires the robot to solve joint configurations using a Newton-Raphson-based Inverse Kinematics (IK) solver [21] at each step. If an unfeasible solution is returned by the IK solver before the robot completes the task, we will assign a negative reward to RL policy. Hence, the adapter learns to optimize the end-effector (EE) trajectory generated by the original RL model. In addition to generalization across robots, we further explore the application of adapter learning techniques between similar tasks. For example, how skills like pulling a drawer can be adapted and transferred to opening a door. We found that this learning method is effective for transferring robotic skill learning.

In summary, in this paper, we explore the following three questions:

- How can adapters be used in robotic reinforcement learning to generalize a global skill across different robotic embodiments?
- What impacts do various adapter architectures and fine-tuning strategies have on skill generalization?
- Does adapter technology have broader application scenarios, such as domain adaptation for similar tasks?

We studied the generalization of the drawer-opening skill across three mobile manipulation robots in the SAPIEN ManiSkill environment [22], including the ManiSkill A2-Single-Arm Robot (A2Single), the Unitree Aliengo robot [23] with Z1 arm [24] (AliengoZ1), and the Toyota Human Support Robot (HSR) [25]. Our research show that adapter learning can effectively generalize among robots with different physical constraints. Particularly, LoRA-type adapters improved the success rate on new robots by 11% on A2Single Arm, 15% on AliengoZ1, and 14% on HSR, compared to vanilla full-finetuning. Our task-variant experiments, including door opening and chair pushing tasks, indicate that adapter learning also improves generalization among tasks.

II. Related Work

A. Learning across robotic embodiments

Given the diverse landscape of robotics, mastering learning across various robotic platforms emerges as a fundamental topic within the field. Based on this concern, a considerable body of work is dedicated to exploring foundational models [7], or delving into the potential of large language models (LLMs) [26]. Additionally, others harness extensive

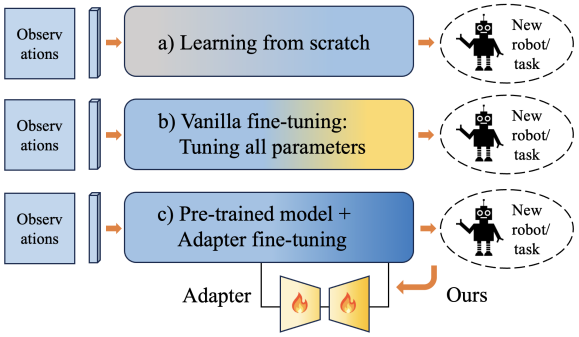


Fig. 2. **Illustration of different approaches.** Commonly seen approaches for new robot/task transfer include learning from scratch and vanilla fine-tuning. In this work, we adopt adapter learning with feedback reward in RL for skill generalization from a pre-trained policy model.

datasets featuring a multitude of robots and demonstrations for comprehensive training approaches, highlighted in the study by the Open X-embodiment [27]. Concurrently, a distinct trajectory in research adopts a structured approach to manipulation policy learning, embracing the hierarchical and modular constructs to bridge the embodiment disparities across varied robotic entities, a notable example being the work by [9]. Nevertheless, these studies often demarcate their focus to either task simplification or specific transfer facets such as 2D kinematic setups [11], end-effector design variants [13], or adaptive visual perception strategies [28]. While there are relevant contributions in adjacent fields such as robotic navigation and locomotion, exemplified by [29] and [30] respectively, these studies tangentially relate to our concentrated objective on robotic manipulation.

B. Generalizable manipulation policy learning

Learning generalizable manipulation skills is a crucial topic in embodied AI research. Numerous works in the visual domain have been proposed to address generalization among objects, including domain-invariant 3D feature distillation [31], 3D affordance learning [32], unified representations of actionable parts [33], and enhancement of both policy and visual modules through interactive perception [34], among others. Additionally, in the domain of policy learning, various works in visual reinforcement learning and imitation learning have been proposed to solve generalization across objects or tasks. For instance, methods that use decoupling or EE action spaces [35], [36] or action primitives [37], [38] have been put forward to increase the efficiency of reinforcement learning and enhance generalizability. On the other hand, modular structures, representational alignment [9], or adversarial generative models [39] have proven effective in generalizing across different objects and tasks. With the advent of large models, recent work has involved integrating Vision-and-Language Models (VLMs) [40] for action and semantic matching, resulting in policies with improved generalizability.

C. Adapter Learning for Policy Model Fine-Tuning

PEFT, such as the adapter technique, is extensively utilized for model domain adaptation. It was first applied in the field of natural language processing; an example being the use of Low-Rank Adaptation (LoRA) [19] for fine-tuning GPT-3 [41], which involves inserting trainable rank decomposition

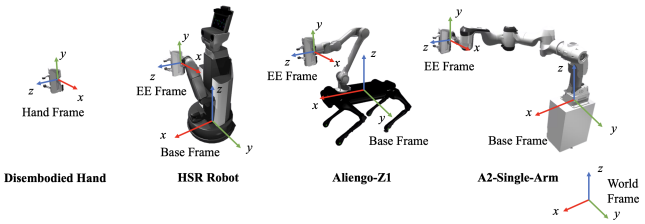


Fig. 3. **Coordinate system of robots and the disembodied hand**. We assess three different mobile manipulators equipped with the same end-effector but featuring varying kinematic configurations.

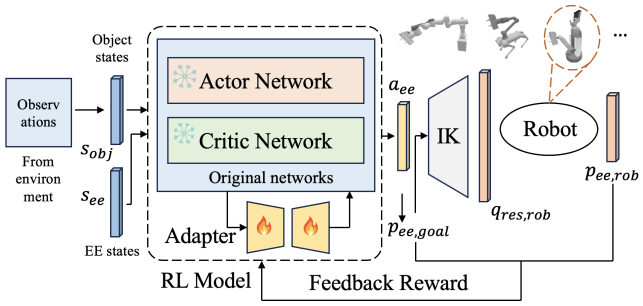


Fig. 4. **Pipeline of our method.** We integrate the adapter module into the RL model and introduce a feedback reward function from the whole-body robotic control. Through this way, the adapter learns to optimize the EE trajectory generated by the original RL model for robot-level generalization.

matrices into each layer. Moreover, adapters are widely adopted in the vision domain [14], [42]–[44]. For instance, ViT-Adapter [42] can achieve performance comparable to that of vision-specific transformers. Adapters are also used in robotic imitation learning, as demonstrated by the work of Liang et al (LoRA-Transformer) [45]. A notable example is TAIL [46], which fine-tunes large, task-specific models. This work is similar to pure vision fields, adapters are used to adjust upstream visual perception modules for manipulation tasks, such as RoboAdapters [47]. Distinct from these applications, our work contemplates the use of adapter technology through RL for generalization across different robotic platforms (as shown in Fig. 2), and we further explore its adaptability across tasks.

III. Methodology

Our methodology focuses on transferring a learned skill from an original robot, e.g., a disembodied hand, to other different robotic platforms, e.g., whole-body mobile manipulators, through the integration of adapters. The adapters serve a dual purpose: fine-tune the model to fit the specific kinematic constraints of a new robot whilst maintaining the integrity of the original learned skill.

A. Problem Statement

Robotic manipulation policy learning can be formulated as a Markov decision process (MDP) [48], which is represented as (S,A,R,T,γ), where S is the set of states, A is the set of actions, $R(s_t,a_t,s_{t+1})$ is the reward function, $T(s_{t+1}|s_t,a_t)$ is the transition function as a probability distribution, and γ is the discount factor for the future rewards. The agent policy $\pi(a|s)$ is the action selecting probability under a given state s. The goal of RL is to maximize the return under the policy $G_\pi = \mathbb{E}_\pi[\sum_t \gamma^t R(s_t,a_t,s_{t+1})]$. In robot learning tasks, we usually need to estimate the task-relevant states from observation O, regarded as $s = f(o)$. This setting is viewed as a partially observable Markov decision process (POMDP) [49] where the policy is $\pi(a|f(o))$.

In this work, we study the problem of generalizing skills across robotic mobile manipulators, where the state space being $s = [s_{obj}, s_{rob}]$ and the action space being $a = [v_{ee}, q_{jaw}]$. We use a shared action space across robots and equip them with the same two-parallel-jaw hand, as shown in Fig. 3.

B. Reinforcement Learning with Disembodied Hand

We first exploit the disembodied hand as the RL agent to learn the abstract skill dynamics, whose DoFs are three virtual prismatic joints and three virtual revolute joints. The action space includes six desired velocities of the virtual joints $v_{ee} \in \mathbb{R}^6$ and two desired positions of the finger joints $q_{f,d} \in \mathbb{R}^2$. For the cabinet environment in ManiSkill, we use $s_{obj} = [s_{cab}, s_{link}, s_{hdl}, s_{size}]$, where s_{cab} is the base link pose of the loaded cabinet, s_{link} and s_{hdl} are the current poses and the full poses (i.e., the poses when the drawer is fully opened) of the target drawer link and the handle, and s_{size} is the full length and the opening length of the target drawer. The poses are all represented as world frame coordinates and quaternions. In RL training, $s_{rob} = s_{ee}$ includes the hand joints' positions and velocities. We follow the dense reward function designed in ManiSkill [22] to train the RL model:

$$R_{ms} = \begin{cases} R_{stg} + R_{ee}, & d > d_{ths}, \\ R_{stg} + R_{ee} + R_{link}, & d < d_{ths},\ c < c_{open}, \\ R_{stg} + R_{ee} + R_{link} + R_{stc}, & d < d_{ths},\ c > c_{open}, \end{cases} \tag{1}$$

where R_{stg} increases from the first stage to the final and the stage is defined by the distance between EE and the handle of the target drawer $d \in \mathbb{R}$ and the opening extent of the target drawer $c_{op} \in [0,1]$. More detailed definitions and coefficients can be found in ManiSkill. The model is trained using soft actor-critic (SAC) [50] algorithm. We view this process as the model pre-training in this work and focus on how to fine-tune the learned policy. The policy is represented as:

$$a_{ee} = \pi_\phi(s_{obj}, s_{ee}). \tag{2}$$

C. Robotic Control Feedback Reward

Given that the RL agent is a disembodied free-floating hand learning the dynamics of skills without considering the constraints of any specific embodied robot control, trajectories that are unfeasible for the robot may occur during skill transfer. Since different robots have different kinematic properties, it is difficult to design a unified constraint condition on the disembodied hand agent without leading to highly limited or sub-optimal trajectories. Therefore, we choose to introduce adapter techniques to fine-tune the original model, which requires us to design a feedback loop to optimize the pre-trained policy network, as shown in Fig. 4.

We first parallel the environment of the disembodied hand with that of the whole-body robot. The action $a_{ee,t}$ executed in the hand environment yields the next pose x_{t+1}. We aim to synchronize the robot's EE to the pose of the disembodied hand, which is expressed as:

$$x_{t+1} = p_{t+1} = p(q_t + \Delta q), \tag{3}$$

where $p(q)$ denotes the forward kinematics equation, and q represents the robot joint position. This desired joint position can be obtained by computing the Jacobian matrix, where:

$$p_{t+1} = p(q_{t+1}) \approx p(q_t) + J(q_t)\Delta q \tag{4}$$

therefore, the approximated value is:

$$\Delta q \approx J^{+}(q_t)\Delta p, \tag{5}$$

where J^{+} is the pseudoinverse of the Jacobian matrix. We use an IK solver based on the Newton-Raphson method to iterate and find a numerical solution Δq_{res} s.t.:

$$p(q_t + \Delta q_{res}) - x_{t+1} < k_{errtol} \tag{6}$$

For robots with high DoFs (considered to be more than 6 here), it is generally not easy to fall into a situation without any IK solutions. However, a global-searching IK solver cannot guarantee a smooth solution between the two EE positions of consecutive time steps. Therefore, we added a restriction to prevent the robot's current configuration from deviating more than k_{maxdev} along each axis. With such a setting, the IK solver iteratively operates through the Jacobian inverse technique. We use the IK-solve-nearby function from the Klampt library [21] to achieve the above setting, represented as:

$$q_{res}, z_{res} = F_{ik}(q, \Delta p, k_{maxdev}, k_{errtol}, k_{maxiter}), \tag{7}$$

where $z_{res} = 1$ when the IK has a feasible solution otherwise $z_{res} = 0$. q_{res} is the solution joint configuration.

For each RL simulation step, we perform the IK calculation for the robot and then control the robot to the solution joint positions before the next RL prediction. Thus, the EE poses that the robot cannot reach through the above IK setting are considered non-compliant with the robot's kinematic constraints, and we use a reward function:

$$R_{ik} = \begin{cases} +1, \ z_{res} = 1, \\ -1, \ z_{res} = 0. \end{cases} \tag{8}$$

This reward varies for different robots, as their kinematic parameters differ, such as joint limits and the structure and number of joints and links. Therefore, the final reward during the fine-tuning process can be expressed as:

$$R = \omega_{ms} R_{ms} + \omega_{ik} R_{ik}. \tag{9}$$

D. Adapter Modules for Parameter Fine-Tuning

When considering fine-tuning the policy network to a specific robot domain, the vanilla approach involves tuning all of the parameters of the original network. However, this method might overfit to the new domain, reducing the model's capability. Adapter technology offers an alternative by introducing a small number of trainable parameters to adjust the original network. A key benefit of this approach is that transferring skills among different robots only requires attaching the corresponding adapter network. This process typically leaves the original network structure and inference speed unaffected. Furthermore, the parameters of the pre-trained policy network can be shared, streamlining the integration process. Common adapter structures include parallel adapters, such as LoRA, and sequential adapters, such as the residual encoder-decoder, which can improve the model's generalizability.

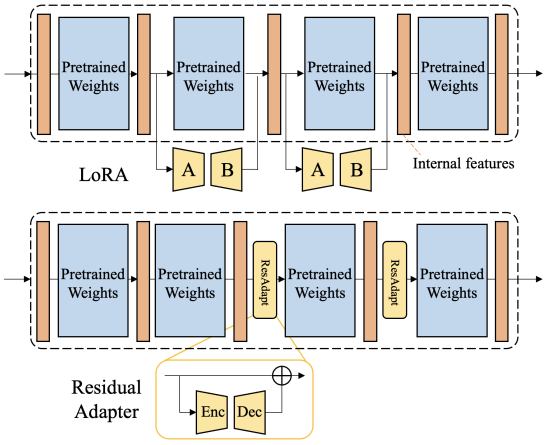


Fig. 5. **Two types of adapters incorporated in this work.** In our implementation, we incorporate the adapters in both the policy (actor) network and the Q-function network in the RL model.

The principle of LoRA is to parallel two low-dimensional matrices in the network's linear layers, represented as A and B, with B initialized to zero. LoRA hypothesizes that the rank of the parameters that need to be adjusted in the original linear layer matrix is likely not high, e.g., 1/10 of the original matrix, so the adjustment can be achieved through learning A and B. Another commonly seen adapter pattern is the encoder-decoder with non-linear activation functions using residual structure (ResAdapter) to connect between original layers. The scale parameter of the residual connection points is initialized to zero. Both structures are widely used to adjust networks based on MLP or Transformer architectures.

In this work, we integrate LoRA or ResAdapter into the linear layers of our MLP, as shown in Fig. 5, and then we use the above reward to finetune the networks. However, we point out that the precise choice of the adapter is not critical to our framework and that our method could relatively easily incorporate other adapters. Since we are using the SAC algorithm, we need to update the parameters of both the actor and critic networks:

$$\begin{aligned} \pi_{\phi_{nn}, \phi_{adapter}} &: \phi_{adapter} \leftarrow \phi'_{adapter}, \\ Q_{\theta_{nn}, \theta_{adapter}} &: \theta_{adapter} \leftarrow \theta'_{adapter}, \end{aligned} \tag{10}$$

where θ represents the critic (Q-function) parameters, and ϕ represents the actor (policy) parameters.

E. Adjusted Adapter and Optimization Techniques for Task-Specific Constraints

In addition to skill generalization across different robots, we further explore more challenging scenarios, namely applying the adapter technique to generalization across tasks. Different tasks typically have distinct trajectories, for instance, as shown in Fig. 6, where pulling a drawer and opening a door are characterized by circular and linear motions respectively, while pushing a chair is usually in the opposite direction to pulling a drawer. Due to significant changes in skill dynamics, we designed a more complex adapter module based on ResAdapter (as shown in Fig. 7) to achieve this adaptation, which can take new inputs and optimize outputs.

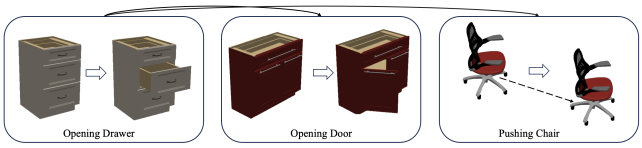


Fig. 6. **Generalization to various tasks.** We demonstrate the adapter technique for generalizing a skill across other manipulation tasks.

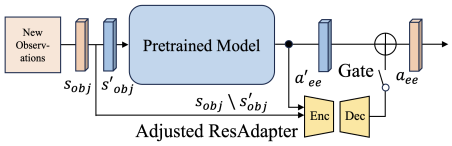


Fig. 7. **Adjusted adapter structure for task-level generalization.** We adjust the structure of ResAdapter to tune the pre-trained model for more challenging scenarios such as new tasks.

To accommodate new tasks, we first align the input states s_{obj} with the corresponding task. For the door opening task, we align the pose of the door handle and door link with the drawer, retaining the original position but leaving the rotation for the Adapter, represented as $\{s_{obj} \setminus s'_{obj}\}$. Additionally, we equate the door's arc length to the drawer's linear length, leaving the door's radius to the Adapter. For the pushing chair task, we map the main body link pose of the chair to the handle and link pose of the drawer but reverse the xy-plane coordinates.

The adjusted residual adapter uses a gate control for the residual connection. The gating signal $d < d_{ths}$ is the distance between the hand and the operational part e.g., the backrest or armrest of the chair. As the new skill dynamics change largely compared to the original skill, in this scenario, we allow the parameters of the original network to be fine-tuned to achieve adaptation to the new states.

IV. Experimental Results

A. Experimental Setups

1) Environments and tasks: We conducted experiments in the SAPIEN ManiSkill simulation environment, choosing the task of opening cabinet drawers as the basic task. The criterion for task success is opening the target joint to $\geq 90\%$ of its extent and that the EE poses are feasible for the whole-body robots. For the drawer-opening task, ManiSkill provides 25 cabinets with different geometries and topologies, of which we randomly select 15 as the training set and 10 as the test set. To evaluate the skill transfer to different robots, the success criteria included the feasibility of the EE poses in the trajectory, which is validated by the Newton-Raphson-based IK solver described in Sec. III-C. We also conducted experiments for task generalization, considering the adaptation of the skill of opening drawers to opening doors and pushing chairs. The success criterion for the door-opening task is to open the specified joint to $\geq \pi/4$ radian. For the pushing chair task, the criterion is that the chair is close to the target position within 0.15 m and remains standing upright. These tasks are also episodic, with a maximum length of 200 steps (each step is 1/20 s).

2) Robotic mobile manipulators: In simulation, we present three mobile manipulation robots, as in Fig. 1:

- **Disembodied Hand**: Modeled as a floating hand with 6-DoF and equipped with two parallel jaws, using the Panda Hand as the collision model.
- **Robot A - A2Single**: Modeled as an 11-DoF robot with a 4-DoF mobile base allowing for x, y, z translations and yaw rotation. It features a Scirus body and a 7-DoF Franka Panda arm.
- **Robot B - HSR**: Modeled as an 8-DoF robot with a 3-DoF mobile base for xy translation and yaw rotation, and it is equipped with a 5-DoF arm.
- **Robot C - AliengoZ1**: Modeled as a 9-DoF robot with a 3-DoF mobile base for xy translation and yaw rotation. Here, we follow the work of Habitat [51], assuming that the base has 3 DoFs of command and ignoring the low-level leg movements. In addition, the robot is fitted with a 6-DoF Z1 robotic arm.

In our real-world experiments, we employ the HSR robot for sim-to-real validation. The outcome and the supplementary videos are available on our project website.

3) Comparison Methods: We compare different methods of generalization to robots: **a) Direct-Transfer**: directly using the model of the trained disembodied hand agent, **b) Full-FineTune**: tuning all the parameters of the original model, **c) ResAdapter**: using the residual adapter with the original model parameters frozen, and **d) LoRA**: using the LoRA adapter with the original model parameters frozen.

B. Effectiveness of adapter learning for multi-robot transfer

Table I shows the average task success rates, episode length in steps, and episode rewards. Comparing Direct-Transfer and LoRA Adapter, we note that introducing robot inverse kinematics control feedback reward can improve the success rate of tasks and accelerate task completion time. This indicates that RL-generated trajectories are optimized such that the introduction of feedback from the specific robot's kinematic constraints improves the success. Meanwhile, by comparing Direct-Transfer and Full-Finetune, we found that if the new scenario differs from the original training scenario, directly fine-tuning a pre-trained model may lead to overfitting. The pre-trained model might already be optimized for the specific characteristics of its original robot, and fine-tuning it on a different robot could cause the model to overly adapt to the new scenario, thus leading to poor generalization on unseen data. Comparing the fine-tuned model performance on different robots, we find that the adapter learning method is more effective for HSR and AliengoZ1 than for A2Single. This might indicate that they have more valid samples, i.e., infeasible poses generate corresponding feedback. Since reward in reinforcement learning is a weak supervisory signal, we consider introducing stronger signals such as auxiliary loss in the future to improve tuning efficiency. Our further experiment on LoRA shown in Fig. 11 illustrates that after applying LoRA on an embodied robot, there is an improvement in the original domain, and it can also be transferred to other robots, demonstrating the effectiveness of the adapter. Meanwhile, the effect of LoRA on the original robot is higher than on other robots, indicating the necessity of applying adapters for each individual robot.

C. Impact of various adapters for skill generalization

Comparing the three methods of fine-tuning, we find that fine-tuning the entire original model based on constraints

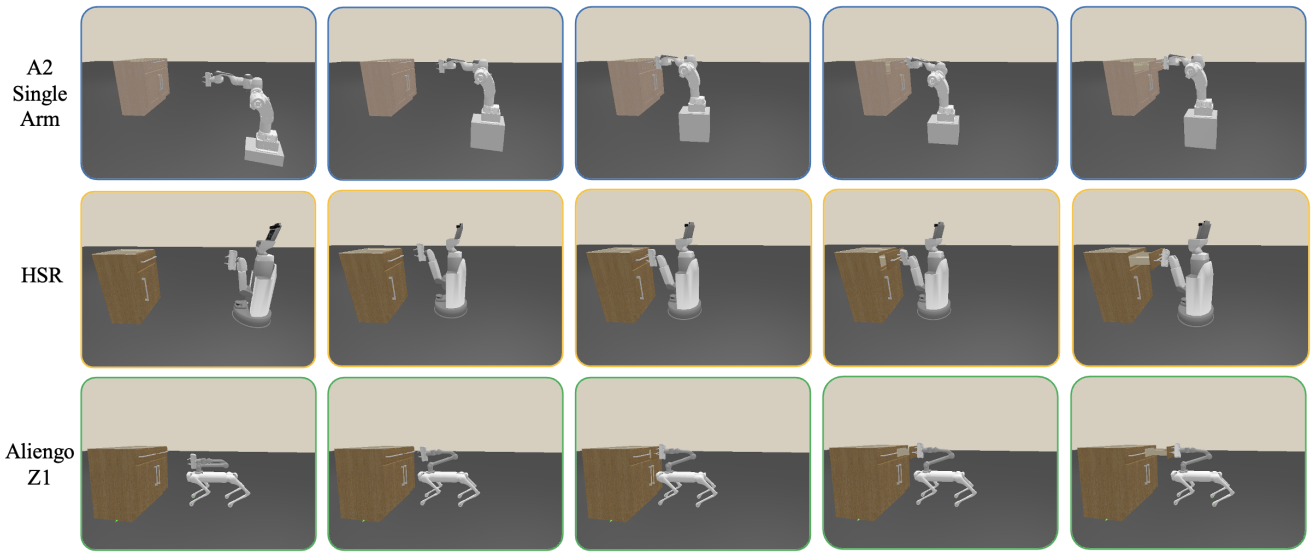


Fig. 8. **Visualization of generalizing a skill across various robotic platforms.**

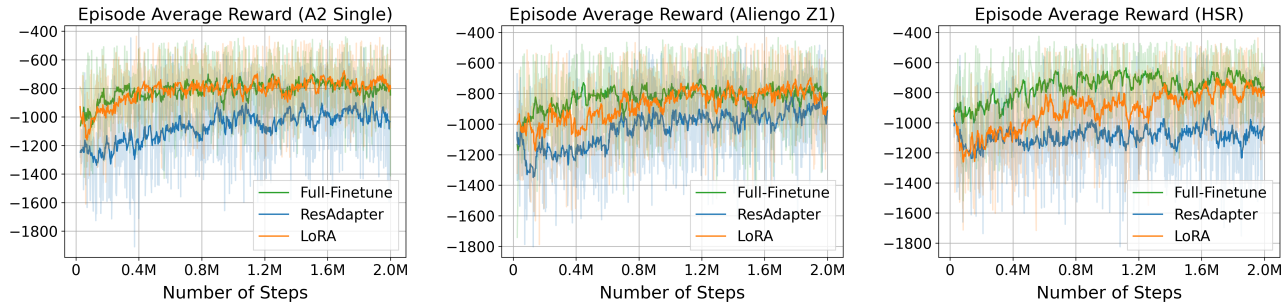


Fig. 9. **Training curve of transferring skill to three different high-DoF robots: A2 Single, Aliengo Z1 and HSR.**

TABLE I
CROSS-ROBOT GENERALIZATION

Robot	Method	Success ↑	Length ↓	Reward ↓
Hand	SAC	68%	115.85	-1084.91
A2Single	Direct-Transfer	32%	155.64	-1500.25
	Full-Finetune	25%	168.19	-1563.99
	LoRA Adapter	**36%**$^{\uparrow 11\%}$	**150.61**	**-1452.61**
AliengoZ1	Direct-Transfer	27%	164.98	-1652.33
	Full-Finetune	22%	169.26	-1702.40
	LoRA Adapter	**37%**$^{\uparrow 15\%}$	**150.85**	**-1503.93**
HSR	Direct-Transfer	26%	163.20	-1508.26
	Full-Finetune	23%	170.51	-1619.22
	LoRA Adapter	**37%**$^{\uparrow 14\%}$	**148.29**	**-1410.24**

presents a high success reward in training as shown in Fig. 9, but relatively lower performance in the test set. This indicates that although it meets the constraints of the new robot, it leads to forgetting the ability to generalize. ResAdapter presents a better performance in the test set, as shown in Fig. 10, by adjusting the intermediate features of the network and deepening it through concatenation. LoRA overall performs the best, indicating that adjusting the parameters of the MLP linear layer in parallel can allow the network to maintain its original good skill generalization ability while meeting the specific kinematic constraints of the robot. We also compare the two adapters' performance on different robots in Table II, and we visualize the task execution by the three robots in Fig. 8, where their policies are fine-tuned by LoRA.

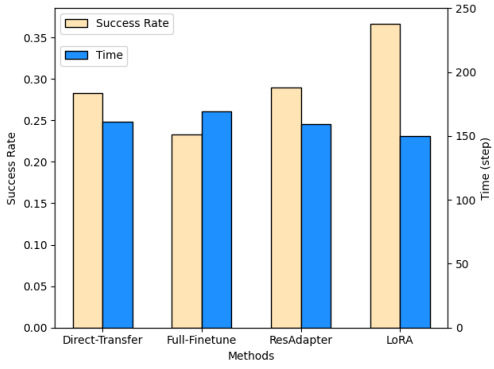
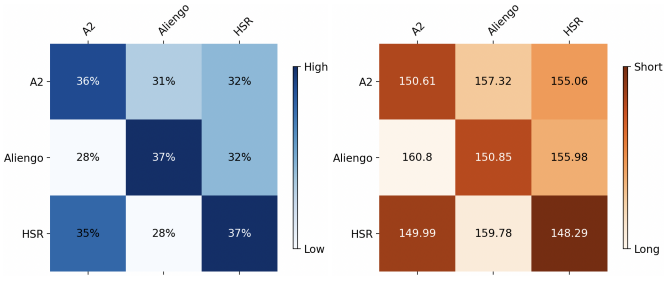


Fig. 10. **Comparison of direct transfer and different tuning methods.** The values are averaged over the three robots.



(a) Task Success Rate (b) Episode length (steps)

Fig. 11. **Cross-robot evaluation.** The adapter is trained on one robot (vertical labels), and tested on the other robot (horizontal labels).

TABLE II

CROSS-ROBOT GENERALIZATION

Robot	Method	Success ↑	Length ↓	Reward ↓
Hand	SAC	68%	115.85	-1084.91
A2Single	ResAdapter	28%	160.18	-1476.89
	LoRA	36%	150.61	-1452.61
AliengoZ1	ResAdapter	27%	162.07	-1558.45
	LoRA	37%	150.85	-1503.93
HSR	ResAdapter	32%	156.35	-1481.21
	LoRA	37%	148.29	-1410.24

TABLE III

CROSS-TASK GENERALIZATION

Robot	Method	Success ↑	Length ↓	Reward ↓
Opening Drawer	SAC	0.68	115.85	-1084.91
Opening Door	Direct-Align	0.	200.00	-2026.08
	Full-Finetune	23%	174.76	-1807.69
	Adapter	**31%**$^{\uparrow 8\%}$	**172.09**	**-1735.80**
Push Chair	Direct-Align	3%	196.61	-3794.65
	Full-Finetune	39%	166.17	-2267.71
	Adapter	**54%**$^{\uparrow 15\%}$	**152.77**	**-1954.63**

D. Adapter-Based Tuning for Cross-Task Generalization

We further explore the generalizability of the adapter technique at the task level, including transfer from opening drawers to opening doors, and from opening drawers to pushing chairs. We consider three methods to generalize the original skill: direct alignment (using the input alignment described in the methodology), fine-tuning the original network, and incorporating adapters for fine-tuning. Fig. 12 shows the training process, where using adapters for input processing and output optimization allows the network to converge faster during training and achieve higher rewards in the door-opening task compared to fine-tuning the original network with new inputs. Table III compares the success rates on the test set, where we find that adapter-based tuning has higher task success rates, indicating that adding adapter modules can enhance the generalizability of the RL policy across tasks. We also notice that transferring the original skill policy to pushing chairs performs better than transferring to doors opening task. This may be because the skill dynamics of the chair are easier to adapt to compared to the originally learned skill dynamics of the network. Fig. 13 shows typical scenarios for the three different tasks, from which we can see that finetuning the pre-trained model can achieve transformations of the skill dynamics.

V. CONCLUSIONS

Our findings indicate that adapter learning is a simple, yet powerful strategy for generalizing robotic skills across different robots and tasks, offering a parameter-efficient alternative to full model retraining. By training first on an entirely virtual disembodied manipulator, the adapter is simply responsible for embodiment-specific adaptation. LoRA-type adapters show improvement in task success rates, supporting their potential for widespread application in robotic learning. Future research may extend these techniques to a broader range of tasks and robots, further enhancing the adaptability and efficiency of robotic systems.

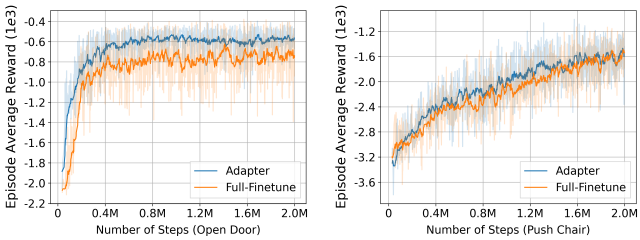


Fig. 12. **Training curves of transferring the opening drawer skill to the opening door task and the pushing chair task.**

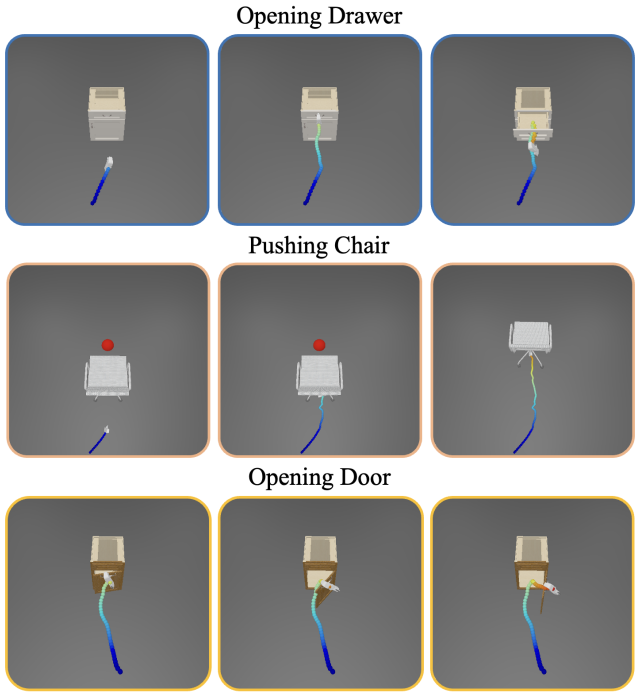


Fig. 13. **Visualization of generalizing a skill across different tasks.**

REFERENCES

[1] Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. In *7th Annual Conference on Robot Learning*, 2023.

[2] Vijaykumar Gullapalli, Judy A Franklin, and Hamid Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, 14(1), 1994.

[3] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms, 2020.

[4] Kaichen Zhou, Shiji Song, Anke Xue, Keyou You, and Hui Wu. Smart train operation algorithms based on expert knowledge and reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(2):716–727, 2020.

[5] Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3, 2019.

[6] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[7] Konstantinos Bousmalis, et al, and Nicolas Heess. RoboCat: A Self-Improving Foundation Agent for Robotic Manipulation, 2023.

[8] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia

Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent, 2022.

[9] Jonathan Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training One Policy Across Robots While Embracing Variability, 2023.

[10] Lin Shao, Fabio Ferreira, Mikael Jorda, Varun Nambiar, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Oussama Khatib, and Jeannette Bohg. UniGrasp: Learning a Unified Model to Grasp With Multifingered Robotic Hands. *IEEE Robotics and Automation Letters*, 5(2):2286–2293, 2020.

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

[12] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control, 2020.

[13] Gautam Salhotra, I.-Chun Arthur Liu, and Gaurav Sukhatme. Learning Robot Manipulation from Cross-Morphology Demonstration, 2023.

[14] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023.

[15] Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun, Ying Zang, and Papa Mao. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3367–3375, 2023.

[16] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[17] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[18] Junliang Guo, Zhirui Zhang, Linli Xu, Boxing Chen, and Enhong Chen. Adaptive adapters: An efficient way to incorporate bert into neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 2021.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv:2106.09685*, 2021.

[20] Jiaxiang Cheng, Pan Xie, Xin Xia, Jiashi Li, Jie Wu, Yuxi Ren, Huixia Li, Xuefeng Xiao, Min Zheng, and Lean Fu. Resadapter: Domain consistent resolution adapter for diffusion models. *arXiv e-prints*, 2024.

[21] Kris Hauser. Klamp't: Kris' Locomotion and Manipulation Planning Toolbox. http://klampt.org, 2022.

[22] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv:2107.14483*, 2021.

[23] Unitree Aliengo. https://www.unitree.com/aliengo.

[24] Unitree Z1. https://www.unitree.com/z1.

[25] Takashi Yamamoto, Tamaki Nishino, Hideki Kajima, Mitsunori Ohta, and Koichi Ikeda. Human support robot (HSR). Association for Computing Machinery, 2018.

[26] Yifan Zhou, Shubham Sonawani, Mariano Phielipp, Simon Stepputtis, and Heni Ben Amor. Modularity through Attention: Efficient Training and Transfer of Language-Conditioned Policies for Robot Manipulation, 2022.

[27] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[28] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real view invariant visual servoing by recurrent control, 2017.

[29] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A General Navigation Model to Drive Any Robot, 2023.

[30] Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My Body is a Cage: The Role of Morphology in Graph-Based Incompatible Control, 2021.

[31] Haoran Geng, Ziming Li, Yiran Geng, Jiayi Chen, Hao Dong, and He Wang. Partmanip: Learning cross-category generalizable part manipulation policy from point cloud observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[32] Kaichun Mo, Leonidas J Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[33] Zhenjia Xu, Zhanpeng He, and Shuran Song. Universal manipulation policy network for articulated objects. *IEEE robotics and automation letters*, 7(2), 2022.

[34] Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects. *arXiv:2106.14440*, 2021.

[35] Kai Lu, Bo Yang, Bing Wang, and Andrew Markham. Decoupling skill learning from robotic control for generalizable object manipulation, 2023.

[36] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh 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)*, pages 1010–1017. IEEE, 2019.

[37] Murtaza Dalal, Deepak Pathak, and Ruslan Salakhutdinov. Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives. *arXiv:2110.15360 [cs]*, October 2021. arXiv: 2110.15360.

[38] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating Reinforcement Learning with Learned Skill Priors. October 2020.

[39] Hao Shen, Weikang Wan, and He Wang. Learning category-level generalizable object manipulation policy via generative adversarial self-imitation learning from demonstrations. *IEEE Robotics and Automation Letters*, 7(4):11166–11173, 2022.

[40] Haoran Geng, Songlin Wei, Congyue Deng, Bokui Shen, He Wang, and Leonidas Guibas. Sage: Bridging semantic and actionable parts for generalizable articulated-object manipulation under language instructions, 2023.

[41] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[42] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[43] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022.

[44] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.

[45] Anthony Liang, Ishika Singh, Karl Pertsch, and Jesse Thomason. Transformer adapters for robot learning. In *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.

[46] Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. Tail: Task-specific adapters for imitation learning with large pretrained models. In *The Twelfth International Conference on Learning Representations*, 2023.

[47] Mohit Sharma, Claudio Fantacci, Yuxiang Zhou, Skanda Koppula, Nicolas Heess, Jon Scholz, and Yusuf Aytar. Lossless adaptation of pretrained vision models for robotic manipulation. In *The Eleventh International Conference on Learning Representations*, 2023.

[48] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2, 1990.

[49] Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*. Springer, 2012.

[50] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 10–15 Jul 2018.

[51] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. *arXiv:2106.14405 [cs]*, June 2021. arXiv: 2106.14405.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning
Publication Status	x Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Kai Lu, Kim Tien Ly, William Hebberd, Kaichen Zhou, Ioannis Havoutis, Andrew Markham. "Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning". *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024.*

Student Confirmation

Student Name:	Kim Tien Ly		
Contribution to the Paper	• Developed the project idea • Wrote the code for the Toyota Human Support Robot • Conducted all real robot experiments • Wrote the paper		
Signature		Date	12/11/2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Ioannis Havoutis			
Supervisor comments I herewith certify that Kim Tien Ly made a substantial contribution to the publication and the contribution described above is accurate.			
Signature		Date	12/11/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

6.3 Discussion

Our framework developed in this chapter, InteLiPlan, aims at enhancing the autonomy and interpretability of domestic robots through the use of LLMs. We design a lightweight, onboard LLM-based system that integrates multimodal inputs such as user commands, visual observations, and motion feasibility checks to generate action plans for mobile manipulation tasks. The key goal of the framework is to enable robots to understand, plan, and replan using human-language commands and guidance. InteLiPlan performs comparably to state-of-the-art robotics planners while being computationally efficient enough for real-time onboard deployment.

A significant feature of InteLiPlan is its HITL mechanism, which allows human intervention when system failures occur. For example, if the robot encounters a vision or motion failure, it asks users for guidance to complete the specified task. This approach ensures greater operational robustness and adaptability in complex domestic environments, where robots might otherwise fail to execute tasks correctly. By enabling human instruction, InteLiPlan makes the system more resilient and responsive to failures in its operating environment.

InteLiPlan is fine-tuned on lightweight LLM and is designed to work in real-time, providing onboard computational capabilities. This makes it suitable for use in everyday domestic settings without relying on external resources. With text-only fine-tuning, the method avoids reliance on robot-specific data, highlighting its adaptability and potential for wide deployment.

The system was evaluated on the simulated and physical HSR in a variety of tasks, verifying its embodied intelligence. By fine-tuning LLMs, the model captures sufficient world knowledge from the foundation model, which helps with generalisation to unseen commands and unseen objects. Meanwhile, the multimodal approach enables the robot to reason around failures and provide contextual information for transparency and interoperability during human-robot interaction. The sampling-based reachability check specifically increases the success rate of executing the generated plan in comparison to the state-of-the-art SayCan [111].

In line with InteLiPlan, the collaboration work in Section 6.2 addresses challenges with adapting manipulation skills to different robot configurations and tasks. The policy directly contributes to the motion library mentioned in the InteLiPlan paper, which emphasizes the technical benefits of generalizable motion planning and highlights how it aligns with the design goals of InteLiPlan. The study provides the InteLiPlan framework in Section 6.1 with an interface where high-level plans can be flexibly executed.

This chapter concludes an LLM-based TAMP solution with the two papers' contributions, showcasing a plug-and-play and generalizable framework. However, in InteLiPlan, we did not implement a validation layer prior to execution. As a result, the robot follows the generated plan regardless of its correctness, and only fails if an action is not found in the list of available motion APIs. Future improvements could include incorporating a verification step before execution to enhance safety and reliability. Besides, though incorporating multiple features in this lightweight planner, the framework relies on human instruction for failure recovery. Further fine-tuning is also required to understand and accommodate new tasks. The research opens new opportunities for incremental learning [138, 139] and low-level motion reactivity in TAMP, which will be discussed in Chapter 7.

7 Conclusion

This thesis studied the problem of task and motion planning for mobile manipulators. The developed projects address the challenges from different perspectives, ranging from hierarchical structures and optimization methods to AI integration with foundation models.

In this last chapter, I will present a critical discussion on what was done, what was learned, and how we can improve them in the future.

7.1 Achievements and Limitations

Firstly, I will discuss the contribution presented in this thesis, complementing the ideas shared at the end of the chapters, and providing the connections between different systems.

Hierarchical structure

In Chapter 4, we developed a two-layer hierarchical structure to autonomously perform multi-surface coverage planning on legged manipulators. The system efficiently decomposes the complexity and optimally solves this complex problem. We extend the planner to a shared-autonomy system which allows user specification in the problem. The perception modules ensure that the framework does not need any prior knowledge about the number of surfaces or surface models of the

inspection sites. Formulating the robot model as a floating-base manipulator allows the integration of the system on different mobile manipulation platforms while ensuring the dexterity of the robot arm.

Using hierarchical structures has been useful in decomposing the complexity of TAMP, whether to break down task and motion levels or to define subgoals for long-horizon planning. When it comes to optimality in TAMP, separating the optimal problems in levels might result in suboptimality of the final solutions. While not guaranteeing global optimal, hierarchical structure is still efficient for solving more detailed task constraints (e.g. coverage planning) while ensuring whole-body motion planning.

Embodied intelligence in TAMP

Chapter 5 similarly solves a TAMP problem with optimization-based approaches. However, the framework in this project scales up to general domestic tasks and introduces more integration between the task planner and the motion planner. The designed reachability graph has improved both symbolic planning and trajectory optimization results in LGP. The main contribution is that using such a heuristic informs the symbolic planner of the physical capability of the robots, which allows the robot to generate reachability-aware decisions and reduces replanning due to low-level feasibility failures. From a wider perspective, this work explored the idea of combining sampling and optimization methods' advantages, which demonstrates competitive performance to the baseline and SOTA algorithms. Additionally, this idea of injecting low-level constraints to symbolic planners also well serves the research in Chapter 6, where we utilized the reachability graph as the feasibility check input to the LLM-based multi-modal planner.

Despite introducing a neat way to incorporate low-level constraints during planning, the R-LGP framework has its own limitations. This work trades off real-time planning with optimality, it also needs pre-defined tasks and assumes known environments while generating path guidance for the full path optimization level.

Foundation models in TAMP

In chapter 6, we trade off the optimality feature of the solution and focus more on a real-time interactive robotics planner. With the capability of understanding human language, LLM has been efficient in capturing the common sense of how the world operates. We utilized this capability to allow adding new tasks without the burden of using classical planning domain languages. In this work, we address onboard computing with a lightweight version of LLM, while maintaining the reasoning capability in smaller models. We chose fine-tuning methods to not only take advantage of the language model but also inject the desired format of the model outcome, which directly executes without translating modules. The multi-modal structure helps the fine-tuned model make decisions that consider relevant components including user command, environment observation, and robot kinematics feasibility. The extensive evaluation has proven the scalability and adaptability over commands and task variations. My collaborative work complements this robotics planner by introducing adapter-based techniques to fine-tune pre-trained skills for generalization, contributing to the development of a motion planner within a plug-and-play TAMP framework.

Being a fine-tuned approach, the planner requires additional fine-tuning to accommodate new tasks or skills, especially those that fall outside the scope of its pre-trained capabilities. There is also no feedback to validate the generated sequence. Though being able to demonstrate the desired interactive behaviours, the robot relies entirely on user feedback for failure recovery and has no autonomous replanning capability. Potential future developments are discussed in Section 7.3.

7.2 Lessons Learned

During my DPhil, I also learned many research and technical lessons on robotic system design and deployment. This section provides a broader perspective on important challenges I recognised while developing my work.

A major challenge in developing TAMP solutions was achieving seamless end-to-end integration across its subsystems. Effective system integration was especially critical in bridging these two levels, which requires specifying inter-dependencies and determining information representation during subproblem communication. TAMP operates on a domain description designed to balance accuracy at its abstraction level with the simplicity needed for efficient planning. This coarse nature needs to be sufficiently translated to condition motion planning. Moreover, TAMP typically requires harmonizing high-level task planning with low-level motion execution, but the process also demands continuous alignment between inputs, such as perception data, and outputs, which include precise actions in real-world scenarios. While our work was not primarily focused on perception, the planning pipeline could not function independently and autonomously without perception input. Low-level control is also important in allowing the successful execution of the motion planning result.

When tackling motion optimization problems in the context of mobile manipulators, I have learned to model the problem by determining cost functions, constraints, and optimization algorithms. Adjusting weights is vital in enabling the robot to balance multiple conflicting objectives, which ensures context-aware planning by considering specific requirements and operational conditions of the robotic system. For example, optimizing for speed and precision often requires carefully balanced tradeoffs, which are controlled by adjusting the weights assigned to the respective cost functions and constraints. Setting thresholds for feasibility, such as acceptable levels of risk or tolerances for positional accuracy in manipulation tasks, is crucial to ensure that the solutions not only meet performance criteria but also adhere to safety standards.

Deploying TAMP solutions on physical robotic systems introduced further experimental issues, particularly due to environmental noise and unexpected physical interactions. For example, when putting an object into the drawer, the robot tends to consider the pose of putting the object down as a collision and decides not to execute the action. Noises from real-world environments also impact sensor readings,

highlighting the necessity of resilient perception modules capable of delivering robust, reliable inputs under variable conditions. Otherwise, this dependency may disrupt the planning process and ultimately affect execution fidelity. This motivated me to address the replanning capability of TAMP in Chapter 6 to recover from these perceptive failures. These findings indicate that TAMP should address perceptual robustness and error mitigation strategies for more reliable and autonomous behaviors.

In addition, distributing onboard computational resources also presents a core challenge for autonomous robots with online planning. LLM-based planners generally require cloud usage to process such large models, hence, also depends on network robustness. In many cases, it became necessary to tradeoff ideal solutions for ones that were practical within real-world constraints. This experience reinforced the importance of designing TAMP solutions with scalability in mind, ensuring that systems can perform effectively in varied conditions while respecting practical hardware limits.

7.3 Future Directions

To conclude this thesis, I present future directions for task and motion planning research that I hope to motivate future projects in the field.

Dual-process TAMP planner

In Chapter 6, we utilized LLM to enable interpretable human-robot interaction that enables replanning upon failures. However, the failure recovery capability depends entirely on human instruction. Future work can consider extending a reactive motion planner, reducing human intervention when encountering failure. Motivated by the dual process theory of human thought similar to [140], the idea is to integrate a reactive layer that deals with common failures without the need of human intervention. Balancing between choosing which level of replanning also poses a challenge in this thread.

Moreover, robust error-handling mechanisms are essential for real-time adaptability, especially in complex or ambiguous scenarios. Addressing these needs suggests that future TAMP frameworks would benefit from more advanced feedback loops that facilitate quick detection of errors and autonomous response, creating a more resilient planning system capable of overcoming minor issues without extensive recalibration. This adaptability, paired with error tolerance, is crucial for reliable TAMP deployment in variable environments and should remain a central focus for further development. For instance, the REMANI-Planner introduces a motion planning method capable of generating high-quality, safe, agile, and feasible trajectories for mobile manipulators in real-time, which is ideal for replanning on the fly in response to unforeseen changes [141].

Incremental Learning

Developing TAMP algorithms that use incremental learning could allow robots to continually improve their understanding of environments and tasks. For instance, a factory robot may need to adapt to slight workspace changes or handle new objects, which could be achieved through continual learning frameworks [139] that retain new knowledge without forgetting previous tasks. Further exploration into incremental learning strategies could improve robots' long-term adaptability and reduce the need for offline reprogramming, enabling autonomous adaptation to varying environments and tasks.

Meta-learning in TAMP could allow robots to generalize skills from previous tasks and apply them to new ones without extensive retraining. For example, a service robot might adapt quickly to new household tasks by leveraging prior knowledge through meta-learning [142]. This approach could reduce the need for task-specific programming, enabling robots to autonomously tackle a wider variety of tasks with minimal setup.

Incorporating HITL learning into TAMP can make robots more responsive to individual user needs and safety requirements by integrating human feedback directly into the learning process. Robots could adjust their plans based on human

demonstrations or cues, learning more effectively in shared environments. For example, Zhu et al. [143] explored combining reinforcement and imitation learning for interactive learning systems, providing a foundation for HITL TAMP. By incorporating human feedback, robots could refine their actions, resulting in more intuitive collaboration and customization in dynamic, user-centered settings.

Human-robot Collaboration

Dynamic task sharing offers a promising avenue for enhancing TAMP in human-robot collaboration by enabling robots to allocate and reallocate tasks seamlessly in response to evolving human actions and environmental changes. This adaptability could be particularly beneficial in both industrial and service environments, where robots might need to step in temporarily if a human is preoccupied or encounters physical limitations. Future research could focus on algorithms that dynamically adjust task allocation based on real-time monitoring of human activity, environmental conditions, and task complexity. For example, Liu et al. [144] introduced a reinforcement learning-based approach for dynamic task-level decision-making, which provides a foundation for further work on adaptive TAMP algorithms that can flexibly assign tasks as circumstances evolve. Building on such foundations, researchers could design TAMP systems that enhance productivity and safety, adapting intelligently within shared spaces.

Intention prediction is another area that could significantly impact TAMP by allowing robots to anticipate human actions and adjust their plans proactively. In shared task settings, accurate intention prediction could lead to smoother task execution and reduced latency, as robots could initiate complementary actions without explicit human commands. This approach could minimize task interruptions, lower the cognitive load on human collaborators, and enhance efficiency and safety [145–147]. In short, human intention prediction would enable robots to engage more fluidly in collaborative tasks, paving the way for more responsive and adaptive TAMP systems.

References

[1] Masoumeh Mansouri, Federico Pecora, and Peter Schüller. "Combining task and motion planning: Challenges and guidelines". In: *Frontiers in Robotics and AI* (2021), p. 133.

[2] Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. "Recent trends in task and motion planning for robotics: A survey". In: *ACM Computing Surveys* 55.13s (2023), pp. 1–36.

[3] Marc Toussaint. "Logic-geometric programming: An optimization-based approach to combined task and motion planning". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[4] Raymond M Smullyan. *First-order logic*. Courier Corporation, 1995.

[5] Jon Barwise. "An introduction to first-order logic". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 90. Elsevier, 1977, pp. 5–46.

[6] Richard E Fikes and Nils J Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving". In: *Artificial intelligence* 2.3-4 (1971), pp. 189–208.

[7] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christianson, et al. "Pddl - the planning domain definition language". In: *Technical Report, Tech. Rep.* (1998).

[8] Levente Kocsis and Csaba Szepesvári. "Bandit based monte-carlo planning". In: *European conference on machine learning*. Springer. 2006, pp. 282–293.

[9] Marc Toussaint. "Newton methods for k-order markov constrained motion problems". In: *arXiv preprint arXiv:1407.0414* (2014).

[10] Qian-Yi Zhou and Vladlen Koltun. "Dense Scene Reconstruction with Points of Interest". In: *ACM Transactions on Graphics* 32 (July 2013).

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Int. Conf. on Knowledge Discovery and Data Mining*. Portland, Oregon: AAAI Press, 1996.

[12] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[13] J Redmon. "You only look once: Unified, real-time object detection". In: (2016).

[14] Takashi Yamamoto, Tamaki Nishino, Hideki Kajima, Mitsunori Ohta, and Koichi Ikeda. "Human support robot (hsr)". In: *ACM SIGGRAPH 2018 emerging technologies*. 2018, pp. 1–2.

[15] Willow Garage. *PR2 (Personal Robot 2).* `http://www.willowgarage.com/pages/pr2/overview`. 2010.

[16] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[17] Maciej Kurant, Athina Markopoulou, and Patrick Thiran. "On the bias of BFS (breadth first search)". In: *2010 22nd International Teletraffic Congress (LTC 22)*. IEEE. 2010, pp. 1–8.

[18] John H Reif. "Depth-first search is inherently sequential". In: *Information Processing Letters* 20.5 (1985), pp. 229–234.

[19] Rina Dechter and Judea Pearl. "Generalized best-first search strategies and the optimality of A". In: *Journal of the ACM (JACM)* 32.3 (1985), pp. 505–536.

[20] Naresh Gupta and Dana S Nau. "On the complexity of blocks-world planning". In: *Artificial intelligence* 56.2-3 (1992), pp. 223–254.

[21] Li-Hong Juang. "Humanoid robot runs maze mode using depth-first traversal algorithm". In: *Multimedia Tools and Applications* 82.8 (2023), pp. 11847–11871.

[22] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[23] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[24] Akshay Kumar Guruji, Himansh Agarwal, and DK Parsediya. "Time-efficient A* algorithm for robot path planning". In: *Procedia Technology* 23 (2016), pp. 144–149.

[25] Sven Mikael Persson and Inna Sharf. "Sampling-based A* algorithm for robot path-planning". In: *The International Journal of Robotics Research* 33.13 (2014), pp. 1683–1708.

[26] Jiansheng Peng, Yiyong Huang, and Guan Luo. "Robot path planning based on improved A* algorithm". In: *Cybernetics and Information Technologies* 15.2 (2015), pp. 171–180.

[27] Anthony Stentz et al. "The focussed D* algorithm for real-time replanning". In: *Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI)*. Vol. 95. 1995, pp. 1652–1659.

[28] Håkan LS Younes and Michael L Littman. "PPDDL1. 0: An extension to PDDL for expressing planning domains with probabilistic effects". In: *Techn. Rep. CMU-CS-04-162* 2 (2004), p. 99.

[29] Maria Fox and Derek Long. "PDDL2. 1: An extension to PDDL for expressing temporal planning domains". In: *Journal of artificial intelligence research* 20 (2003), pp. 61–124.

[30] Daniel L Kovacs et al. "A multi-agent extension of PDDL3. 1". In: *ICAPS 2012 Proceedings of the 3rd Workshop on the International Planning Competition (WS-IPC 2012)*. 2012, pp. 19–37.

[31] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 30. 2020, pp. 440–448.

[32] Vladimir Lifschitz. "Answer set programming and plan generation". In: *Artificial Intelligence* 138.1-2 (2002), pp. 39–54.

[33] Yu-qian Jiang, Shi-qi Zhang, Piyush Khandelwal, and Peter Stone. "Task planning in robotics: an empirical comparison of PDDL-and ASP-based systems". In: *Frontiers of Information Technology & Electronic Engineering* 20 (2019), pp. 363–373.

[34] Earl D Sacerdoti. "Planning in a hierarchy of abstraction spaces". In: *Artificial intelligence* 5.2 (1974), pp. 115–135.

[35] Thorsten Belker, Martin Hammel, and Joachim Hertzberg. "Learning to optimize mobile robot navigation based on HTN plans". In: *2003 IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE. 2003, pp. 4136–4141.

[36] Stéphane Magnenat, Jean-Cédric Chappelier, and Francesco Mondada. "Integration of online learning into HTN planning for robotic tasks". In: *2012 AAAI Spring Symposium Series*. 2012.

[37] Oliver Obst. "Using a planner for coordination of multiagent team behavior". In: *International Workshop on Programming Multi-Agent Systems*. Springer. 2005, pp. 90–100.

[38] L-J Lin. "Hierarchical learning of robot skills by reinforcement". In: *IEEE International Conference on Neural Networks*. IEEE. 1993, pp. 181–186.

[39] Bradley Hayes and Brian Scassellati. "Autonomously constructing hierarchical task networks for planning and human-robot collaboration". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 5469–5476.

[40] Alejandro Suárez-Hernández, Guillem Alenyà, and Carme Torras. "Interleaving hierarchical task planning and motion constraint testing for dual-arm manipulation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4061–4066.

[41] Robert Goldman. "A semantics for HTN methods". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 19. 2009, pp. 146–153.

[42] Bo Zhang, Guobin Li, Qixin Zheng, Xiaoshan Bai, Yu Ding, and Awais Khan. "Path planning for wheeled mobile robot in partially known uneven terrain". In: *Sensors* 22.14 (2022), p. 5217.

[43] Thushara Sandakalum and Marcelo H Ang Jr. "Motion planning for mobile manipulators—a systematic review". In: *Machines* 10.2 (2022), p. 97.

[44] Alireza Rastegarpanah, Hector Cruz Gonzalez, and Rustam Stolkin. "Semi-autonomous behaviour tree-based framework for sorting electric vehicle batteries components". In: *Robotics* 10.2 (2021), p. 82.

[45] Ander Iriondo, Elena Lazkano, Loreto Susperregi, Julen Urain, Ane Fernandez, and Jorge Molina. "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning". In: *Applied Sciences* 9.2 (2019), p. 348.

[46] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. "Capturing robot workspace structure: representing robot capabilities". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee. 2007, pp. 3229–3236.

[47] Abhijit Makhal and Alex Goins. "Reuleaux: Robot Base Placement by Reachability Analysis". In: Jan. 2018, pp. 137–142.

[48] Jun Dong and Jeffrey C Trinkle. "Orientation-based reachability map for robot base placement". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 1488–1493.

[49] Wolfgang Merkt, Yiming Yang, Theodoros Stouraitis, Christopher E Mower, Maurice Fallon, and Sethu Vijayakumar. "Robust shared autonomy for mobile manipulation with continuous scene monitoring". In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. IEEE. 2017, pp. 130–137.

[50] Yiming Yang, Vladimir Ivan, Zhibin Li, Maurice Fallon, and Sethu Vijayakumar. "iDRM: Humanoid motion planning with realtime end-pose selection in complex environments". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 271–278.

[51] Neel Dhanaraj, Yeo Jung Yoon, Rishi Malhan, Prahar M Bhatt, Shantanu Thakar, and Satyandra K Gupta. "A mobile manipulator system for accurate and efficient spraying on large surfaces". In: *Procedia Computer Science* 200 (2022), pp. 1528–1539.

[52] Qiankun Yu, Guolei Wang, Xiaotong Hua, Simin Zhang, Libin Song, Jiwen Zhang, and Ken Chen. "Base position optimization for mobile painting robot manipulators with multiple constraints". In: *Robotics and Computer-Integrated Manufacturing* 54 (2018), pp. 56–64.

[53] C Dario Bellicoso, Koen Krämer, Markus Stäuble, Dhionis Sako, Fabian Jenelten, Marko Bjelonic, and Marco Hutter. "Alma-articulated locomotion and manipulation for a torque-controllable robot". In: *2019 International conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 8477–8483.

[54] Parker Ewen, Jean-Pierre Sleiman, Yuxin Chen, Wei-Chun Lu, Marco Hutter, and Ram Vasudevan. "Generating continuous motion and force plans in real-time for legged mobile manipulation". In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 4933–4939.

[55] Henrique Ferrolho, Vladimir Ivan, Wolfgang Merkt, Ioannis Havoutis, and Sethu Vijayakumar. "Roloma: Robust loco-manipulation for quadruped robots with arms". In: *Autonomous Robots* 47.8 (2023), pp. 1463–1481.

[56] Wolfgang Merkt, Vladimir Ivan, Yiming Yang, and Sethu Vijayakumar. "Towards shared autonomy applications using whole-body control formulations of locomanipulation". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 1206–1211.

[57] Foudil Abdessemed. "Trajectory generation for mobile manipulators using a learning method". In: *2007 Mediterranean Conference on Control & Automation*. IEEE. 2007, pp. 1–6.

[58] Yiming Yang, Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar. "Planning in time-configuration space for efficient pick-and-place in non-static environments with temporal constraints". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 1–9.

[59] Giuseppe Oriolo and Marilena Vendittelli. "A control-based approach to task-constrained motion planning". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 297–302.

[60] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[61] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. "Semantic attachments for domain-independent planning systems". In: *Towards Service Robots for Everyday Environments: Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments* (2012), pp. 99–115.

[62] Andreas Hertle, Christian Dornhege, Thomas Keller, and Bernhard Nebel. "Planning with semantic attachments: An object-oriented view". In: *ECAI 2012*. IOS Press, 2012, pp. 402–407.

[63] Caelan Garrett, Tomás Lozano-Pérez, and Leslie Kaelbling. "Sample-based methods for factored task and motion planning". In: (2017).

[64] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. "Combined task and motion planning through an extensible planner-independent interface layer". In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 639–646.

[65] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "Backward-forward search for manipulation planning". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 6366–6373.

[66] Nils J. Nilsson. "Shakey the robot". In: *Technical note 323* (1984).

[67] Stephane Cambon, Rachid Alami, and Fabien Gravot. "A hybrid approach to intricate motion, manipulation and task planning". In: *The International Journal of Robotics Research* 28.1 (2009), pp. 104–126.

[68] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrated Task and Motion Planning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 265–293.

[69] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. "Combined task and motion planning through an extensible planner-independent interface layer". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 639–646.

[70] Neil Dantam, Swarat Chaudhuri, and Lydia Kavraki. "The Task Motion Kit". In: *IEEE Robotics & Automation Magazine* PP (May 2018), pp. 1–1.

[71] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. "Incremental task and motion planning: A constraint-based approach." In: *Robotics: Science and systems*. Vol. 12. Ann Arbor, MI, USA. 2016, p. 00052.

[72] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. "An incremental constraint-based framework for task and motion planning". In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1134–1151.

[73] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. "Manipulation Planning with Probabilistic Roadmaps". In: *The International Journal of Robotics Research* 23.7-8 (2004), pp. 729–746.

[74] Hui X Li and Brian C Williams. "Generative Planning for Hybrid Systems Based on Flow Tubes". In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2008, pp. 206–213.

[75] Cornelius V Braun, Joaquim Ortiz-Haro, Marc Toussaint, and Ozgur S Oguz. "RHH-LGP: Receding Horizon And Heuristics-Based Logic-Geometric Programming For Task And Motion Planning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 13761–13768.

[76] Danny Driess, Ozgur Oguz, and Marc Toussaint. "Hierarchical task and motion planning using logic-geometric programming (HLGP)". In: *RSS Workshop on Robust Task and Motion Planning*. 2019.

[77] An T Le, Philipp Kratzer, Simon Hagenmayer, Marc Toussaint, and Jim Mainprice. "Hierarchical Human-Motion Prediction and Logic-Geometric Programming for Minimal Interference Human-Robot Tasks". In: *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE. 2021, pp. 7–14.

[78] Marc Toussaint and Manuel Lopes. "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4044–4051.

[79] Jan Kristof Behrens, Ralph Lange, and Masoumeh Mansouri. "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8705–8711.

[80] Kyle EC Booth, Tony T Tran, Goldie Nejat, and J Christopher Beck. "Mixed-integer and constraint programming techniques for mobile robot task planning". In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 500–507.

[81] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*. Vol. 271. Springer, 2014.

[82] Daniel Ioan, Ionela Prodan, Sorin Olaru, Florin Stoican, and Silviu-Iulian Niculescu. "Mixed-integer programming in motion planning". In: *Annual Reviews in Control* 51 (2021), pp. 65–87.

[83] Stephen Boyd and Jacob Mattingley. "Branch and bound methods". In: *Notes for EE364b, Stanford University* (2007), pp. 2006–07.

[84] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. "Cutting planes in integer and mixed integer programming". In: *Discrete Applied Mathematics* 123.1-3 (2002), pp. 397–446.

[85] Robin Deits and Russ Tedrake. "Footstep planning on uneven terrain with mixed-integer convex optimization". In: *2014 IEEE-RAS international conference on humanoid robots*. IEEE. 2014, pp. 279–286.

[86] Preston Culbertson, Saptarshi Bandyopadhyay, and Mac Schwager. "Multi-robot assembly sequencing via discrete optimization". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6502–6509.

[87] Martina Lippi and Alessandro Marino. "A mixed-integer linear programming formulation for human multi-robot task allocation". In: *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE. 2021, pp. 1017–1023.

[88] Wil Thomason, Marlin P Strub, and Jonathan D Gammell. "Task and Motion Informed Trees (TMIT*): Almost-Surely Asymptotically Optimal Integrated Task and Motion Planning". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11370–11377.

[89] Sergio Jiménez, Tomás De La Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. "A review of machine learning for automated planning". In: *The Knowledge Engineering Review* 27.4 (2012), pp. 433–467.

[90] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[91] Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Adversarial actor-critic method for task and motion planning problems using planning experience". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8017–8024.

[92] Natasha Balac, Daniel M Gaines, and Doug Fisher. "Learning action models for navigation in noisy environments". In: *ICML Workshop on Machine Learning of Spatial Knowledge, Stanford*. 2000.

[93] Philipp Kratzer, Simon Bihlmaier, Niteesh Balachandra Midlagajni, Rohit Prakash, Marc Toussaint, and Jim Mainprice. "Mogaze: A dataset of full-body motions that includes workspace geometry and eye-gaze". In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 367–373.

[94] Dimitris Bertsimas and Bartolomeo Stellato. "The voice of optimization". In: *Machine Learning* 110.2 (2021), pp. 249–277.

[95] Dimitris Bertsimas and Bartolomeo Stellato. "Online mixed-integer optimization in milliseconds". In: *INFORMS Journal on Computing* (2022).

[96] Bob Bixby. "The gurobi optimizer". In: *Transp. Re-search Part B* 41.2 (2007), pp. 159–178.

[97] Abhishek Cauligi, Preston Culbertson, Bartolomeo Stellato, Dimitris Bertsimas, Mac Schwager, and Marco Pavone. "Learning mixed-integer convex optimization strategies for robot planning and control". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 1698–1705.

[98] Niklas Funk, Svenja Menzenbach, Georgia Chalvatzaki, and Jan Peters. "Graph-based Reinforcement Learning meets Mixed Integer Programs: An application to 3D robot assembly discovery". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 10215–10222.

[99] Prateek Gupta, Elias B Khalil, Didier Chetélat, Maxime Gasse, Yoshua Bengio, Andrea Lodi, and M Pawan Kumar. "Lookback for Learning to Branch". In: *arXiv preprint arXiv:2206.14987* (2022).

[100] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. "Learning to branch in mixed integer programming". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[101] Jonathan Cacace, Riccardo Caccavale, Alberto Finzi, and Vincenzo Lippiello. "Interactive plan execution during human-robot cooperative manipulation". In: *IFAC-PapersOnLine* 51.22 (2018), pp. 500–505.

[102] Ioan A Şucan and Lydia E Kavraki. "Accounting for uncertainty in simultaneous task and motion planning using task motion multigraphs". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 4822–4828.

[103] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Integrated task and motion planning in belief space". In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1194–1227.

[104] Aidan Curtis, Leslie Kaelbling, and Siddarth Jain. "Task-Directed Exploration in Continuous POMDPs for Robotic Manipulation of Articulated Objects". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3721–3728.

[105] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. "Large language models for robotics: A survey". In: *arXiv preprint arXiv:2311.07226* (2023).

[106] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "PDDL Planning with Pretrained Large Language Models". In: *NeurIPS 2022 Foundation Models for Decision Making Workshop*. 2022.

[107] Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Lior Horesh, Biplav Srivastava, Francesco Fabiano, and Andrea Loreggia. "Plansformer: Generating Symbolic Plans using Transformers". In: *NeurIPS 2023 Workshop on Generalization in Planning*. 2023.

[108] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. "Code as policies: Language model programs for embodied control". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9493–9500.

[109] Mengdi Xu, Wenhao Yu, Peide Huang, Shiqi Liu, Xilun Zhang, Yaru Niu, Tingnan Zhang, Fei Xia, Jie Tan, and Ding Zhao. "Creative Robot Tool Use with Large Language Models". In: *NeurIPS 2023 Foundation Models for Decision Making Workshop*. 2023.

[110] Yutao Ouyang, Jinhan Li, Yunfei Li, Zhongyu Li, Chao Yu, Koushil Sreenath, and Yi Wu. "Long-horizon Locomotion and Manipulation on a Quadrupedal Robot with Large Language Models". In: *arXiv preprint arXiv:2404.05291* (2024).

[111] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. "Do as i can, not as i say: Grounding language in robotic affordances". In: *arXiv preprint arXiv:2204.01691* (2022).

[112] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. "Text2motion: From natural language instructions to feasible plans". In: *Autonomous Robots* 47.8 (2023), pp. 1345–1365.

[113] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. "Inner Monologue: Embodied Reasoning through Planning with Language Models". In: *Conference on Robot Learning*. PMLR. 2023, pp. 1769–1782.

[114] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. "Flamingo: a visual language model for few-shot learning". In: *Advances in neural information processing systems* 35 (2022), pp. 23716–23736.

[115] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. "Pali: A jointly-scaled multilingual language-image model". In: *arXiv preprint arXiv:2209.06794* (2022).

[116] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models". In: *International conference on machine learning*. PMLR. 2023, pp. 19730–19742.

[117] Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, et al. "An introduction to vision-language modeling". In: *arXiv preprint arXiv:2405.17247* (2024).

[118] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[119] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation". In: *Conference on robot learning*. PMLR. 2022, pp. 894–906.

[120] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *Conference on Robot Learning*. PMLR. 2023, pp. 2165–2183.

[121] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. "π_0: A Vision-Language-Action Flow Model for General Robot Control". In: *arXiv preprint arXiv:2410.24164* (2024).

[122] Zhutian Yang, Caelan Reed Garrett, Tomás Lozano-Pérez, Leslie Kaelbling, and Dieter Fox. "Sequence-based plan feasibility prediction for efficient task and motion planning". In: *arXiv preprint arXiv:2211.01576* (2022).

[123] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. "PaLM-E: An Embodied Multimodal Language Model". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 8469–8488.

[124] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. "Palm: Scaling language modeling with pathways". In: *Journal of Machine Learning Research* 24.240 (2023), pp. 1–113.

[125] Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. "Physically grounded vision-language models for robotic manipulation". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 12462–12469.

[126] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. "Challenges and applications of large language models". In: *arXiv preprint arXiv:2307.10169* (2023).

[127] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. "Large language models for human-robot interaction: A review". In: *Biomimetic Intelligence and Robotics* (2023), p. 100131.

[128] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. "ChatGPT for Robotics: Design Principles and Model Abilities". In: *IEEE Access* (2024).

[129] Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Pen Xu, Leila Takayama Takayama, Fei Xia, Jake Varley, et al. "Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners". In: *Conference on Robot Learning (CoRL)*. 2023.

[130] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. "Rt-1: Robotics transformer for real-world control at scale". In: *Robotics: Science and Systems* (2023).

[131] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. "Interactive language: Talking to robots in real time". In: *IEEE Robotics and Automation Letters* (2023).

[132] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9118–9147.

[133] Tom B Brown. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[134] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: `2307.09288 [cs.CL]`.

[135] Kim Tien Ly, Matthew Munks, Wolfgang Merkt, and Ioannis Havoutis. "Asymptotically Optimized Multi-Surface Coverage Path Planning for Loco-Manipulation in Inspection and Monitoring". In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2023, pp. 1–7.

[136] Kim Tien Ly, Valeriy Semenov, Mattia Risiglione, Wolfgang Merkt, and Ioannis Havoutis. "R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 14917–14923.

[137] Kai Lu, Kim Tien Ly, William Hebberd, Kaichen Zhou, Ioannis Havoutis, and Andrew Markham. "Learning Generalizable Manipulation Policy with Adapter-Based Parameter Fine-Tuning". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024.

[138] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges". In: *Information fusion* 58 (2020), pp. 52–68.

[139] Rahaf Aljundi, Markus Würger, Abhinav Gupta, Andreas Leonetti, Thiemo Wiedemeyer, Sankalp Arora, and Andrea Vezzani. "Online Continual Learning with No Task Boundaries". In: *arXiv preprint arXiv:1902.10486* (2019).

[140] Thomas Anthony, Zheng Tian, and David Barber. "Thinking fast and slow with deep learning and tree search". In: *Advances in neural information processing systems* 30 (2017).

[141] Chengkai Wu, Ruilin Wang, Mianzhi Song, Fei Gao, Jie Mei, and Boyu Zhou. "Real-time Whole-body Motion Planning for Mobile Manipulators Using Environment-adaptive Search and Spatial-temporal Optimization". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 1369–1375.

[142] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 1126–1135.

[143] Yuke Zhu, Daniel Gordon, Dmitry Berenson, Pieter Abbeel, and Anca D. Dragan. "Robosuite: A Simulation Framework and Benchmark for Robot Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, p. 8959320.

[144] Zhihao Liu, Quan Liu, Lihui Wang, Wenjun Xu, and Zude Zhou. "Task-level decision-making for dynamic and stochastic human-robot collaboration based on dual agents deep reinforcement learning". In: *The International Journal of Advanced Manufacturing Technology* 115.11 (2021), pp. 3533–3552.

[145] Kim Tien Ly, Mithun Poozhiyil, Harit Pandya, Gerhard Neumann, and Ayse Kucukyilmaz. "Intent-aware predictive haptic guidance and its application to shared control teleoperation". In: *2021 30th IEEE international conference on robot & human interactive communication (RO-MAN)*. IEEE. 2021, pp. 565–572.

[146] Weitian Wang, Rui Li, Yi Chen, and Yunyi Jia. "Human intention prediction in human-robot collaborative tasks". In: *Companion of the 2018 ACM/IEEE international conference on human-robot interaction*. 2018, pp. 279–280.

[147] Marc Dalmasso, Jose Enrique Domínguez-Vidal, Iván J Torres-Rodríguez, Pablo Jiménez, Anaís Garrell, and Alberto Sanfeliu. "Shared Task Representation for Human–Robot Collaborative Navigation: The Collaborative Search Case". In: *International Journal of Social Robotics* 16.1 (2024), pp. 145–171.