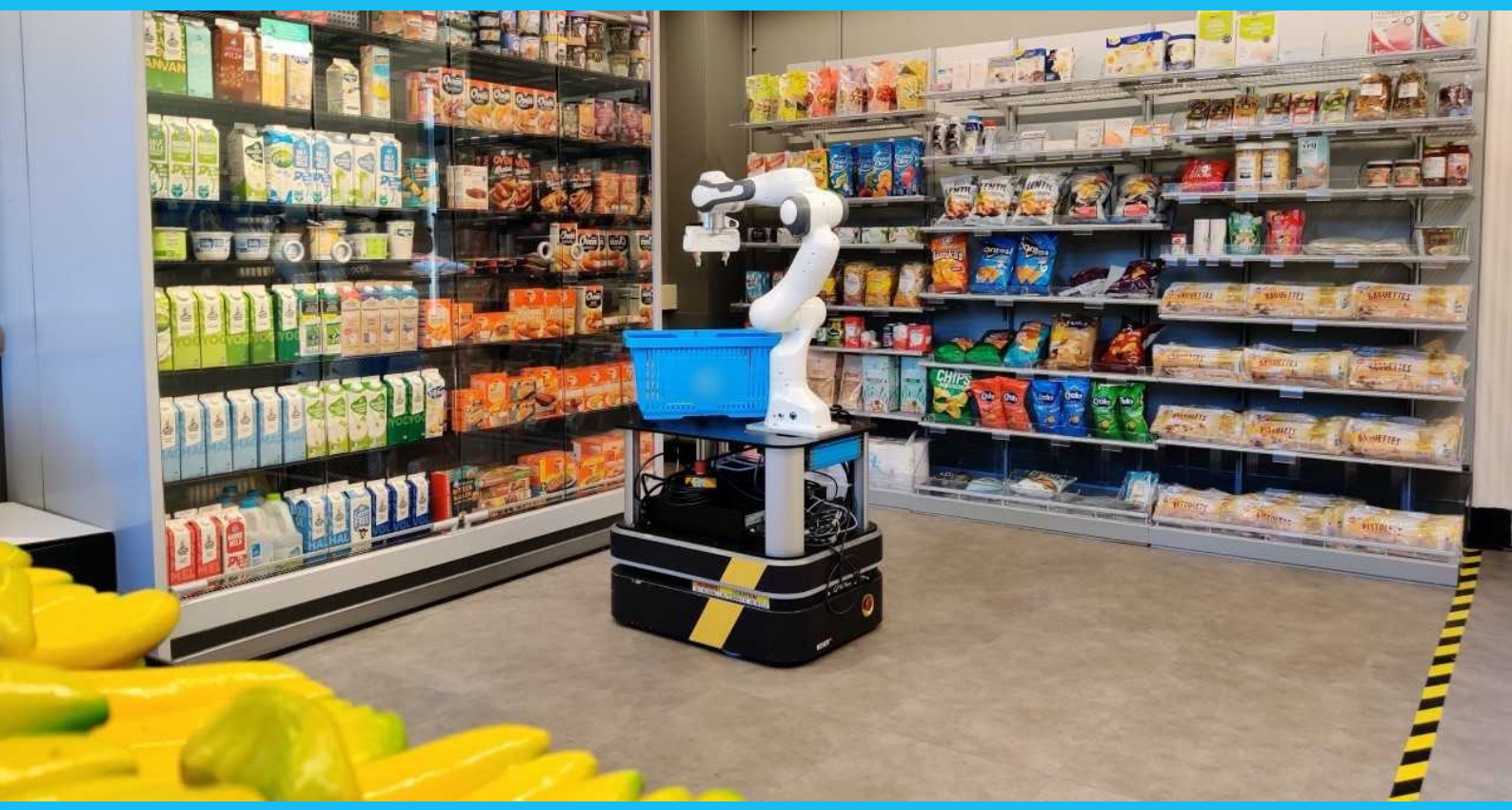
Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning

Thesis Report

Yuezhe Zhang



TU Delft

Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning

by

Yuezhe Zhang

to obtain the degree of Master of Science in Robotics
at the Delft University of Technology,
to be defended publicly on Wednesday August 30, 2023 at 15:30.

Student number: 5390664
Project duration: May, 2022 – August, 2023
Thesis committee:

Dr. J. Alonso-Mora,	TU Delft, main supervisor, chair
Ir. E. Trevisan,	TU Delft, daily supervisor
Ir. C. Pezzato,	TU Delft, daily supervisor
Dr. J. Kober,	TU Delft
Dr. R. Ferrari,	TU Delft

Cover: Image by AIRLab Delft
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

In one of my favorite Cantonese movies *Days of Being Wild*, there is one line, *in our world, there exists a unique bird that has no feet, it can only fly relentlessly, and sleeps in the wind when it is tired*. If this is true, I think the bird is me. For the last decade, during my middle school, high school, and bachelor's, I am always in different cities. The adventure continues until I am now in the Netherlands, where I am fortunate to meet so many nice people and compose a new story.

My deepest gratitude goes to my thesis supervisor, Dr. Javier Alonso-Mora, without whom the story would be very different. Thank you for guiding me to see the beautiful landscape of motion planning, for giving me the opportunity to do my thesis at the Autonomous Multi-Robots Lab, and for the support, discussion, and feedback during the whole thesis. I am also grateful to my two daily supervisors, Ir. Corrado Pezzato and Ir. Elia Trevisan. You are like my big brothers, always there caring for me, helping me tramp over mountains and conquer the hardships and obstacles during the journey. I am also honored to have Dr. Jens Kober and Dr. Riccardo Ferrari on the thesis committee. This will be a great opportunity for me to learn from our discussions and your feedback.

No one can whistle a symphony, it takes an orchestra to play it. That orchestra is lucky to have one in AIRLab. Thank you to Prof. Martijn Wisse for having me join the weekly discussion for the project. Thank you to Chadi for the support and help during the real-world experiments and debugging. You make life much easier when encountering magic reality. Thank you to Max for your passion for open-source coding, which also inspires me essentially. Thank you to Maxi for the lesson, playing ping pong is even harder than doing research. Thank you all for giving me a space where I feel at home. I learned a lot from you in doing research, playing ping pong, and all the games and fun we had.

I never feel proud of myself being any special, but I do take pride in having connected with so many friends. I am so lucky to have Yigubigu as a family around me. Our lives, happiness, and sorrow are interleaved from the very moment we land at the Schiphol airport. I am also lucky to be part of 2358 for the unforgettable Spanish trips and fun we had when playing Mahjong, tennis, and having parties. I am happy to have you guys from the DCF football team for the matches we have gone through and for the beer we had. Witnessing Argentina capture the World Cup Championship with you is one of the most thrilling moments in my life. I also want to thank the teammates and friends I have met during the courses and honours program for the wonderful time in discussing and having fun.

I also want to thank the great Leo Messi and Kobe Bryant for your humbleness, extraordinary skills, and strong mentality, which have always pushed and inspired me to shape my personality and achieve my dreams for the last decades.

Last but not least, my deepest love goes to my parents and families. Thank you so much for bringing me to this splendid world, for paving the road for me when I achieve my dreams, and for your sacrifices and unconditional love, which I could never repay.

This marks the end of my master's study. But the bird will continue to fly until it feels tired or it never feels tired!

Yuezhe Zhang
Delft, August 2023

Abstract

Task and Motion Planning (TAMP) has progressed significantly in solving intricate manipulation tasks in recent years, but the robust execution of these plans remains less touched. Particularly, generalizing to diverse geometric scenarios is still challenging during execution. In this work, we propose a reactive TAMP method to deal with disturbances and geometric ambiguities by combining an active inference planner (AIP) for online action selection with a proposed multi-modal model predictive path integral controller (M3P2I) for low-level control. The AIP generates online alternative plans, each of which is translated into a cost function to be sampled for the proposed method. The proposed M3P2I then uses a parallelizable physics simulator for throwing different rollouts, leading to a coherent optimal solution by averaging the weighted samples based on their costs. Our method empowers real-time adaptation of action sequences to rectify failed plans, while also computing low-level motions to address dynamic obstacles or disturbances that could potentially invalidate the existing plan.

Theoretical findings are validated in simulation and in the real world. We show that the framework exhibits reactiveness in different scenarios, including battery charging, push-pull among obstacles, and pick-place with disturbances. We show that our framework outperforms an off-the-shelf RL method in the reactive pick-place task in terms of position error and orientation error. We also show that M3P2I is generalizable in combining different constraints, such as generating hybrid motions of push and pull for the mobile robot, and grasping objects with different grasping poses. The real-world experiments show that the system exhibits reactiveness and robustness against human disturbances in a variety of manipulation tasks.

Nomenclature

List of Abbreviations

EFE	Expected Free Energy
FEP	Free-Energy Principle
GPU	Graphics Processing Unit
LHS	Left Hand Side
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral Control
POMDP	Partially Observable Markov Decision Process
PRM	Probabilistic Roadmap
RHS	Right Hand Side
RL	Reinforcement Learning
RRT	Rapidly-Exploring Random Tree
TAMP	Task and Motion Planning
VFE	Variational Free Energy

List of Symbols

$s_\tau \in \mathbb{R}^m$	Hidden state at time τ, where m is the number of mutually exclusive values a state can have
$\boldsymbol{s}_\tau^\pi \in \mathbb{R}^m$	Posterior distribution of the hidden state under a plan π, where m is the number of mutually exclusive values a state can have
$o_\tau \in \mathbb{R}^r$	Observation at time τ, where r is the number of values an observation can have
$\boldsymbol{o}_\tau^\pi \in \mathbb{R}^r$	Posterior distribution of observation at time τ, where r is the number of values an observation can have
a_τ	Symbolic action to be performed at time τ
$\pi \in \mathbb{R}^p$	Plan specified by a sequence of actions, where p is the number of actions
$\boldsymbol{\pi} \in \mathbb{R}^p$	Posterior distribution over plans, where p is the number of actions

$\boldsymbol{A} \in \mathbb{R}^{r \times m}$	Likelihood matrix, mapping from hidden states to observations. The columns are categorical distribution $P(o_\tau \vert s_\tau, \boldsymbol{A}) = Cat(\boldsymbol{A} s_\tau)$
$\boldsymbol{B}_{a_\tau} \in \mathbb{R}^{m \times m}$	Transition matrix, indicating the probability of state transition under certain action a_τ. The columns are categorical distribution $P(s_{\tau+1} \vert s_\tau, a_\tau) = Cat(\boldsymbol{B}_{a_\tau} s_\tau)$
$\boldsymbol{C} \in \mathbb{R}^r$	Prior preferences over observation $P(o_\tau) = \boldsymbol{C}$
$\boldsymbol{D} \in \mathbb{R}^m$	Prior probability or belief over initial states $P(s_0) = Cat(\boldsymbol{D})$
$F(\pi) \in \mathbb{R}$	Variational free energy
$\boldsymbol{F}_\pi \in \mathbb{R}^p$	$\boldsymbol{F}_\pi = (F(\pi_1), F(\pi_2), \ldots, F(\pi_p)))^\top$
$G(\pi, \tau) \in \mathbb{R}$	Expected free energy
$\boldsymbol{G}_\pi \in \mathbb{R}^p$	$\boldsymbol{G}_\pi = (G(\pi_1), G(\pi_2), \ldots, G(\pi_p)))^\top$
σ	Softmax function

Contents

List of Figures

List of Algorithms

1 Introduction

This introductory chapter focuses on the motivation behind this thesis, narrowing down the research scope, and posing the fundamental problem we want to address. Subsequently, a brief overview of the fields related to the research topic is provided. The chapter then summarizes the contributions of the thesis. Finally, the outline of the document is given.

1.1. Motivation

The promising future of the automation industry requires robots to use a variety of skills to perceive the environment, navigate in it and perform different tasks. The deployment of these robots could increase overall production, efficiency, and quality in the workplace and could also help human workers in dealing with dangerous or economically infeasible tasks. Especially for retail stores as shown in Figure 1.1, mobile manipulators can reduce the workload of store employees by packaging products, stocking shelves, and cleaning floors. This would in turn lead to improvements in the quality of customer service. However, in order to solve these real-world tasks, the robots should operate safely and efficiently in dynamic and uncertain environments.

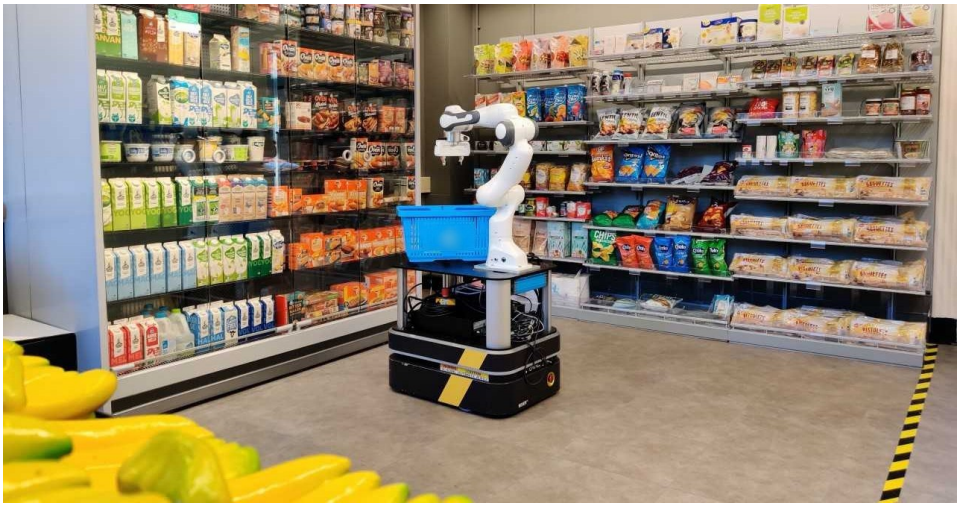


Figure 1.1: A mobile manipulator from the AI for Retail (AIR) Lab Delft [1]

This requires sufficiently advanced technology including perceiving the environment through the sensors, creating high-level plans based on the reasoning about the environment, and then efficiently controlling the actuators to carry out the plans. In this work, the focus is on the latter two aspects,

namely task planning and motion planning, and the interdependence in between.

Task and motion planning is a powerful class of methods for solving complex long-term manipulation problems where logic and geometric variables are influencing each other. TAMP [2] has been successfully applied in many domains such as table rearrangement, stacking blocks, or solving the Hanoi puzzle. Despite impressive recent results [3], the environments in which TAMP plans are executed are usually not dynamic and the plan is executed in open-loop [4]. Recent works [5–7] recognized the importance of robustifying the execution of TAMP plans in order to be able to reliably carry them out in the real world. However, these works either rely only on the adaptation of the action sequence in a plan [7–10] or only on the motion planning problem in a dynamic environment given a fixed plan [5, 6]

A key challenge in reactive TAMP is the handling of geometric constraints that might not be known at planning time. For instance, moving a block to a desired location may necessitate pulling rather than pushing if the block is situated in a corner. This is challenging since the outcome of pushing and pulling actions can be hard to predict accurately even assuming full knowledge of the scene at planning time. Planning a sequence of push-pull actions a priori and then executing it is also prone to fail. Another scenario involves the classic pick-and-place task with sequential actions: reaching the cube, closing the gripper, lifting the cube to a pre-grasp position, and releasing the cube. However, distinct grasping poses are necessary for diverse locations, it is challenging to abstract a generalizable and robust grasping pose constraint capable of accommodating various locations while considering the fact that dynamic obstacles or human disturbances might invalidate initially planned grasping poses.

These research gaps bring to the formulation of the questions that this work aims to address:

- *How can Task and Motion Planning (TAMP) incorporate reactive behaviors from both high-level action selection and low-level motions in multi-task, contact-rich and dynamic environments?*
- *How to address geometric ambiguities, which are hard to determine at planning time, within a reactive TAMP framework?*

1.2. The state-of-the-art

This section provides a brief overview of the fields that entail the topic of the thesis.

1.2.1. Task Planning

Task planning community has focused on planning in discrete domains using representations and algorithms that exploit underlying patterns in the large state space for a long period. Ghallab et al. [11] provides a comprehensive guide to task planning from the AI perspective. Karpas and Magazzeni [12] review task planning in the context of planning robot actions. In this survey, the main effort is to review the robotics-relevant topics of task planning, especially concentrating on generating action plans, while the basic methods of knowledge representation and symbolic reasoning in the classical sense of artificial intelligence behind the scene are omitted.

However, the majority of the task planning literature paid too much attention to developing offline searching techniques, while underestimating the importance of the deliberation capabilities needed to carry out the actions. Authors in [13, 14] pinpointed this issue and advocated changing the focus to the roles of *actors* instead of *planners*. Actors here should not be mere action executors, instead, they should be able to take intelligent decisions and adapt their behaviors to the dynamically changing environment online. This is especially beneficial when mobile manipulators work in dynamic environments, where actions planned offline are prone to fail. More specifically, the actors should possess two properties, as summarized in [13–15]:

- Hierarchical deliberation: each action in a plan may be a task that may need further refinement. The hierarchical deliberation should go beyond the scope of the current hierarchical planning techniques so that the different modules should be integrated effectively.

- Continual online planning and reasoning: the actor will monitor, refine, extend, update, change, and repair its plans throughout the acting process and generate activities dynamically at run-time.

To achieve such actors, various scholars have conducted extensive research on understanding human intelligence in decision-making. One of the most influential theories in neuroscience is the so-called active inference, which tries to explain how information is processed by the human brain [16]. The mathematical framework of active inference was constructed on the Free-Energy Principle (FEP) proposed by Karl Friston [17]. It has shown its potential in the domain of adaptive control such as controlling manipulators [18] and fault tolerant systems [19, 20] and also in the domain of symbolic reasoning [21]. The main idea of Friston's neuroscience theory is that the brain's cognition and motor control functions can be described as *free energy minimization* so that agents can select those actions that maximize the "well-being" and minimize the "surprise" based on Bayesian inference and gradient descent schemes. The brain will also maintain an internal (generative) model that can incorporate the sensor data and update the belief as a posterior in an approximated Bayesian framework. *Active inference suits our case considering its appealing aspects, namely the efficiency in continual online planning and the flexibility in encoding parameterized habits and preferences.*

1.2.2. Motion Planning

The problem of motion planning was formulated by Lozano-Pérez [22] as a search for paths in the robot's configuration space from one state to another without colliding with any obstacles. LaValle [23] provides an overall introduction to the field of motion planning. Motion planning methods can be categorized into two subgroups, global motion planning which computes a trajectory from the initial to the goal configuration directly, and local motion planning which generates a feasible trajectory within short time periods at a higher frequency. The most popular, general, and effective methods are based on sampling [24–26] or constrained optimization [27–29]. Besides, machine learning techniques [30, 31] have also been applied to generate a motor skill trajectory.

Recently, the Model Predictive Path Integral (MPPI) control algorithm has emerged as a powerful and efficient approach that shows promising performance in real-time non-linear dynamics and high dimensional robotic systems [32–35]. The key idea of MPPI is to sample control sequences from a given distribution and forward simulate them in parallel (e.g. on a GPU) using the system's dynamics to generate trajectories. Each simulated trajectory is then evaluated against a cost function. The cost of each trajectory is then converted to importance sampling weights, which will be used to update the cost-weighted average control sequences. However, the clarity of dynamic modeling of the problem remains a challenge, and there is no open-source implementation available. *In general, the aforementioned works have been applied for single-skill execution, such as pushing or reaching a target point, and never in the context of a longer task that requires sequential decision-making.*

Interestingly, [36] presents an open-source application based on MuJoCo physics for real-time predictive control. Among the challenges and future work, the authors suggested the use of learned policies when rolling out samples, and the use of a high-level agent to set the cost function of the predictive controller for long-horizon cognitive tasks. *We follow this line of thought and propose a method to perform the composition of cost functions for long-horizon tasks in a reactive fashion.*

1.2.3. Integrated Task and Motion Planning

Considering that high-level actions could vary depending on the low-level motions and execution of the motions also rely on the high-level actions, especially in dynamically changing environments, high-level actions and low-level motions should be formulated in an integrated way. This introduces the problem of *Task and Motion Planning* (TAMP). The problem of TAMP can be described as a robot taking actions in environments containing many objects to achieve some goals and changing the states of the objects. It contains components of the aforementioned discrete task planning and continuous motion planning, and thus it should be considered as a hybrid discrete-continuous search problem. A review of classical TAMP can be found in [37].

Classical TAMP approaches can solve complex integrated task and motion tasks, but they are not designed to generate reactive behaviors against changing environments and perturbations. It might be true some of them can achieve reactive properties via online re-planning, but this is computationally inefficient if re-planning occurs too often and the task to solve is too complicated. The reactive properties are essential in our research problem considering that the robots will be operated in a dynamic environment which may involve unexpected disturbances and events. In recent years, the TAMP communities have also focused on creating robust, reusable, and reactive behaviors for manipulation tasks, which can be called Reactive TAMP.

In [5], the authors provide a reactive MPC strategy to execute a TAMP plan as a given linear sequence of constraints. Instead of composing primitive skills, [5] derive the control law from a composition of constraints for MPC. The reactive nature of the approach allows coping with disturbances and dynamic collision avoidance during the execution of a TAMP plan. The work in [6] formulates a TAMP plan in object-centric Cartesian coordinates, showing how this allows coping with perturbations such as moving a target location. Both [5, 6], however, do not consider adaptation at the symbolic action level if a perturbation invalidates the current plan.

On the other hand, a number of papers focused on adapting and repairing high-level action sequences during execution. Authors in [38] propose to represent robot task plans as robust logical-dynamical systems. The method can effectively adapt the logic plan in order to deal with external human disturbances. Similarly, [39] presented a method to coordinate control chains and achieved robust plan execution through plan switching and controller selection. [10] proposed instead to blend task and action planners by dynamically expanding a behavior tree at runtime through back-chaining. Similarly, [40] presents a reactive task allocation framework based on behavior trees and linear temporal logic for multi-robot systems. In [7], behavior trees and linear temporal logic have been combined into a reactive TAMP method against a cooperative or adversarial human operator which might invalidate the current plan. A recent work [41] proposes to use Monte Carlo Tree Search in combination with Isaac Gym to speed up task planning for multi-step object retrieval from clutter, where complex physical interaction is required. This is a promising direction, but [41] only focuses on high level reasoning and executes pre-defined motions in open-loop.

In [8], active inference and behavior tree were combined to provide reactive action selection in long-term tasks in partially observable and dynamic environments. This method achieves hierarchical deliberation and continual online planning, which makes it particularly appealing for the problem of reactive TAMP at hand. *In this thesis, we extend [8] via bridging the gap to low-level reactive control by planning cost functions instead of symbolic actions.*

1.3. Contributions

This thesis presents main contributions in combining active inference and MPPI for reactive TAMP, and in proposing a novel Multi-Modal MPPI to address geometric ambiguities. They can be detailed as follows.

- We present a method for reactive TAMP that uses active inference to plan cost functions to be minimized by MPPI instead of symbolic actions. This enables reactive execution from both high-level actions and low-level motions since the cost functions can capture both the task objectives and constraints.
- We propose a Multi-Modal MPPI (M3P2I) that is capable of sampling different alternatives to achieve a given goal and evaluating them against different costs. This enables a coherent optimal solution considering potential plans instead of relying on complex heuristics to switch between these plans.

In addition, a minor contribution is the following.

- We propose a metric for efficiently measuring the difference between two orientations of fully symmetric objects.

1.4. Outline

The thesis is organized as follows.

Chapter 2 provides the necessary knowledge of the two building blocks of the thesis: 1) active inference, a reactive action plan that generates symbolic actions; 2) MPPI, a sampling-based model predictive controller.

Chapter 3 entails the main body of this thesis, a scientific paper titled "*Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning*". We intend to submit it to the IEEE Robotics and Automation Letters (RA-L). The supporting videos can be found at https://sites.google.com/view/m3p2i-aip.

Chapter 4 supports the scientific paper with additional information. It first introduces the overall software structure of this work, and then elaborates on the implementation details and results of an additional experiment.

Chapter 5 presents the discussion about key aspects of the simulation and real-world experiments.

The final chapter summarizes the contents this work presents and the conclusions are made about the original questions this survey aims to answer. Potential future directions closely tied to this work are also delineated.

2

Background

This chapter provides the necessary knowledge of the two building blocks of the thesis: 1) active inference, a reactive task planner that generates symbolic actions; 2) MPPI, a sampling-based model predictive controller. We first introduce the free-energy principle in subsection 2.1.1, which was connected to the Bayesian brain hypothesis to achieve hierarchical deliberation and online reasoning. We then introduce the mathematical description for planning with active inference in subsection 2.1.2, which builds upon the free-energy principle. Subsequently, we present a general discrete time control scheme and an information-theoretic interpretation of optimal control in subsection 2.2.1, which will derive the update law of MPPI. We proceed with the importance sampling technique and the complete algorithm for information-theoretic MPPI in subsection 2.2.2. Finally, discussions will be made about the comparison between the state-of-the-art methods with the presented methods.

2.1. Active Inference

2.1.1. Free-Energy Principle

Free-energy principle was proposed by Karl Friston[17] to explain how information is processed by the human brain, which is an ideal technique for our application with regard to hierarchical deliberation and continual online reasoning. [42] clarifies the connections between FEP and earlier unifying ideas such as Bayesian inference, predictive coding, and active learning in a more comprehensive way.

Resisting a Tendency Towards a Disorder

The motivation of the free-energy principle is that adaptive biological or artificial systems tend to naturally resist disorder, and any such system that is at equilibrium with the environment must minimize its free energy, which is an information theory measure limiting the surprise or improbable outcome. The surprise in information theory describes the atypicality of an event and can be quantified by using the negative log-probability of its sensory data:

$$-\ln p(o) \tag{2.1}$$

where $p(o)$ is the probability of observing particular sensory data o in the environment. The more improbable an event is, the higher the surprise.

From a biological perspective, the ability of biological systems to maintain their states in the face of both external and internal rapidly changing environments can be reduced to their homeostasis. And the states where an agent can be are limited and these define the agent's phenotype. Mathematically speaking, the probability of these exteroceptive and interoceptive sensory states must have a low entropy, which means less surprise, and therefore the agent has to minimize the long-term average of a surprise to ensure that the sensory states remain within physiological bounds. However, an agent cannot know whether its sensations are surprising and cannot avoid them even if it did know. Therefore,

the free energy comes in as an upper bound on surprise. This means if an agent minimizes the free energy, it simply minimizes surprise.

Unrestricted Free-Energy Principle

Mathematically, it is also necessary to introduce free energy. Due to the fact that posterior is typically intractable and it is computationally difficult to search over all possible posterior distributions, the key idea of the FEP is to convert the Bayesian inference into an optimization problem. This idea was first developed in physics and later in machine learning to handle computationally intractable inference problems.

Assume a family of distributions Q are available and one auxiliary distribution $q \in Q$ can be chosen to approximate $p(s|o)$, this leads to the variational optimization problem:

$$q^*(s) = \underset{q(s)}{\text{argmin}}\, KL[q(s)||p(s|o)] \tag{2.2}$$

It should be noticed that the KL divergence is always non-negative due to Jensen's inequality, if $p(s|o)$ is contained in the variational family Q, then the optimum can be achieved with $q^*(s) = p(s|o)$. If the variational family Q contains all possible distributions, it is called unrestricted. The optimization problem can be reformulated in a more computationally practical way, based on the fact that:

$$\begin{aligned} KL[q(s)||p(s|o)] &= \sum_s q(s) \log \frac{q(s)}{p(s|o)} \\ &= \sum_s q(s) \log \frac{q(s)p(o)}{p(s,o)} \\ &= \sum_s q(s) \left[\log \frac{q(s)}{p(s,o)} + \log p(o)\right] \\ &= \sum_s q(s) \log \frac{q(s)}{p(s,o)} + \log p(o) \sum_s q(s) \\ &= F[q(s)] + \log p(o) \geq 0 \end{aligned} \tag{2.3}$$

where the Variational Free Energy (VFE) is defined as:

$$F[q(s)] = \sum_s q(s) \log \frac{q(s)}{p(s,o)} \tag{2.4}$$

The free energy is equivalent to the negative of the *evidence lower bound*, which is common in the machine learning literature [43]. It can be derived from Equation 2.3 that the free energy is the upper bound of the negative log probability of sensory data, which is exactly the surprise that has been defined in Equation 2.1:

$$F[q(s)] \geq -\log p(o) \tag{2.5}$$

Note that the free energy only requires the knowledge of $p(s,o)$, which is easy to compute given the prior $p(s)$ and likelihood $p(o|s)$ of any state. Thus minimizing the free energy will also minimize the surprise. In addition, minimizing the free energy of unrestricted variational family is equivalent to exact Bayesian inference.

Restricting the Variational Family

When the variational family Q does not contain the posterior $p(s|o)$, FEP is no longer equivalent to Bayesian inference, and thus the distribution minimizing free energy will deviate from Bayes-optimality: $q^*(s) \neq p(s|o)$. In order to make the optimization tractable, approximations have to be made for the auxiliary density $q(s)$.

The widely used "mean field" approximation assumes that the variables of s are statistically independent of each other and thus the density can be factorized, which can be used to approximate the posterior:

$$p(s|o) \approx q(s) = \prod_i q_i(s_i) \tag{2.6}$$

When s is continuous, another common approximation, which is called *Laplace approximation*, assumes the distribution is Gaussian, specified by a mean μ and covariance matrix Σ:

$$q(s) \sim \mathcal{N}(s; \mu, \Sigma) \tag{2.7}$$

2.1.2. Active Inference

Inheriting from the free-energy principle, Active Inference was proposed by Karl Friston for describing and explaining how adaptive systems (biological or artificial) conduct perception, learning, planning, and decision making [16, 44]. Active inference uses free energy to describe the properties of an agent in an environment, and by minimizing expected free energy at run time, Bayes-optimal behavior can be obtained. This section provides the necessary mathematical descriptions for planning and decision making with active inference, and a detailed derivation can be found in [45].

Generative Model

In active inference, the real world is described internally as a simplified generative model, which can be framed as a Markov Decision Process to infer the environmental states, to predict the actions and observations. The generative model can be expressed as a joint probability distribution $P(\overline{o}, \overline{s}, \boldsymbol{\eta}, \pi)$, where $\overline{o}$ is a sequence of observations, $\overline{s}$ is a sequence of states, $\boldsymbol{\eta}$ is the required model parameters, and π is a plan. Using the chain rule, the joint probability can be written as:

$$P(\overline{o}, \overline{s}, \boldsymbol{\eta}, \pi) = P(\overline{o}|\overline{s}, \boldsymbol{\eta}, \pi) P(\overline{s}|\boldsymbol{\eta}, \pi) P(\boldsymbol{\eta}|\pi) P(\pi) \tag{2.8}$$

Since the sequence of observations $\overline{o}$ is conditionally independent of the parameters $\boldsymbol{\eta}$ and the plan π given $\overline{s}$ and the Markov property guarantees that the next state and current observation only depend on the current state, it can be written as:

$$P(\overline{o}|\overline{s}, \boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(o_\tau|s_\tau) \tag{2.9}$$

Considering that the states $\overline{s}$ and parameters $\boldsymbol{\eta}$ are conditionally independent given plan π, it can be simplified as:

$$P(\overline{s}|\boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi) \tag{2.10}$$

The probability of parameters $\boldsymbol{\eta}$ given plan π can be described as:

$$P(\boldsymbol{\eta}|\pi) = P(\boldsymbol{A}) P(\boldsymbol{B}) P(\boldsymbol{D}) \tag{2.11}$$

where the parameters represent the following meanings: $\boldsymbol{A}$ is the likelihood matrix, indicating the probability of observations given a specific state; $\boldsymbol{B}$ is the transition matrix, indicating the possibility of state transition under certain action. For specific a_τ, $\boldsymbol{B}_{a_\tau}$ represents the probability of state $s_{\tau+1}$ after applying action a_τ from state s_τ; $\boldsymbol{D}$ defines the probability about the initial state. Each columns of $\boldsymbol{A}$, $\boldsymbol{B}_{a_\tau}$, $\boldsymbol{D}$ is a categorical distribution $Cat()$. Combining Equation 2.8, Equation 2.9, Equation 2.10, Equation 2.11, the generative model can be derived as:

$$P(\overline{o}, \overline{s}, \boldsymbol{\eta}, \pi) = P(\overline{o}, \overline{s}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \pi) = P(\pi) P(\boldsymbol{A}) P(\boldsymbol{B}) P(\boldsymbol{D}) \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi) P(o_\tau|s_\tau) \tag{2.12}$$

Variational Free Energy

The Variational Free Energy (VFE) is used to measure the fit between the internal generative model and past and current sensory information. It can be expressed as:

$$F(\pi) = \sum_{\tau=1}^{T} s_\tau^{\pi\top} \left[\ln s_\tau^\pi - \ln \boldsymbol{B}_{a_{\tau-1}} s_{\tau-1}^\pi - \ln \boldsymbol{A}^\top \boldsymbol{o}_\tau \right] \tag{2.13}$$

where $F(\pi)$ is a plan-specific free energy and the logarithm operation is considered element-wise.

Perception

Perception and learning can be interpreted as minimizing the variational free energy. The posterior distribution of the state given a plan can be expressed as:

$$\begin{aligned} s^{\pi}_{\tau=1} &= \sigma\left(\ln \boldsymbol{D} + \ln \boldsymbol{B}^{\top}_{a_\tau} s^{\pi}_{\tau+1} + \ln \boldsymbol{A}^{\top} o_\tau\right) \\ s^{\pi}_{1<\tau<T} &= \sigma\left(\ln \boldsymbol{B}_{a_{\tau-1}} s^{\pi}_{\tau-1} + \ln \boldsymbol{B}^{\top}_{a_\tau} s^{\pi}_{\tau+1} + \ln \boldsymbol{A}^{\top} o_\tau\right) \\ s^{\pi}_{\tau=T} &= \sigma\left(\ln \boldsymbol{B}_{a_{\tau-1}} s^{\pi}_{\tau-1} + \ln \boldsymbol{A}^{\top} o_\tau\right) \end{aligned} \tag{2.14}$$

where σ is the softmax function and the columns of $\ln \boldsymbol{B}^{\top}_{a_\tau}$ are normalized.

Expected Free Energy

An agent updates beliefs about future states which can be used to calculate the Expected Free Energy (EFE). The expected free energy is necessary to evaluate alternative plans. Plans that lead to preferred observations are more likely to be chosen. Preferred observations are specified in the model parameter $\boldsymbol{C}$. The expected free energy for a plan π at time τ is given by:

$$G(\pi, \tau) = o^{\pi\top}_{\tau} \left[\ln o^{\pi}_{\tau} - \ln C\right] - diag\left(A^{\top} \ln A\right) s^{\pi}_{\tau} \tag{2.15}$$

By minimizing expected free energy, the agent balances reward term $o^{\pi\top}_{\tau} \left[\ln o^{\pi}_{\tau} - \ln C\right]$ and information seeking term $diag\left(A^{\top} \ln A\right) s^{\pi}_{\tau}$.

Planning and Decision Making

Planning and decision making can be interpreted as minimizing the expected free energy. The posterior distribution over p possible plans is obtained by:

$$\boldsymbol{\pi} = \sigma\left(-\boldsymbol{G}_{\pi} - \boldsymbol{F}_{\pi}\right) \tag{2.16}$$

where $\boldsymbol{F}_{\pi} = (F(\pi_1), F(\pi_2), \ldots, F(\pi_p)))^{\top}$ and $\boldsymbol{G}_{\pi} = (G(\pi_1), G(\pi_2), \ldots, G(\pi_p)))^{\top}$.

Given the aforementioned posterior distribution and the plan dependent states s^{π}_{τ}, the overall probability distribution for the states can be computed through Bayesian Model Average:

$$s_\tau = \sum_i s^{\pi_i}_{\tau} \boldsymbol{\pi}_i, i \in \{1, \ldots, p\} \tag{2.17}$$

where $s^{\pi_i}_{\tau}$ is the probability of a state at time τ under plan i and $\boldsymbol{\pi}_i$ is the probability of plan i.

Finally, the action for the agent to execute is the action with the highest probability:

$$\begin{aligned} \lambda &= \text{argmax}\left([\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_p]\right) \\ a_\tau &= \pi_\lambda(\tau = 1) \end{aligned} \tag{2.18}$$

where λ is the index of the most likely plan.

The active inference algorithm [45] can be summarised as pseudo-code in Algorithm 1. The algorithm starts by setting the prior preferences over observations. Then it enters the loop by getting observations and computing the VFE for each plan. The posterior state is then updated to compute the EFE for each plan. Once the terms of free energy are computed, we can obtain the posterior distribution over p possible plans and the overall probability distribution for the states. Finally, the action with the highest probability is selected among the p possible plans.

Algorithm 1 Action selection with active inference [45]

Set $\boldsymbol{C}$ ▷ Prior preferences
for $\tau = 1$ to T **do**
  If not specified, get state from $\boldsymbol{D}$ if $\tau == 1$
  If not specified, get observation from $\boldsymbol{A}$
  Compute F for each plan ▷ Equation 2.13
  Update posterior state s_τ^π ▷ Equation 2.14
  Compute G for each plan ▷ Equation 2.15
  Bayesian model averaging ▷ Equation 2.17
  Action selection ▷ Equation 2.18
end for
Return a ▷ Preferred action

2.2. Model Predictive Path Integral Control

Generalization always remains a major challenge for intelligent decision making and fast execution, especially when the robots work in dynamic and uncertain environments and thus must react to new situations. Model Predictive Control (MPC) [46–49] addresses this problem via constrained optimization in a receding horizon way. However, most MPC methods rely on convexification of the constraints and cost functions, which is inflexible for high-dimensional systems such as mobile manipulation. In addition, manipulation tasks often involve discontinuous contact and complex cost terms which are hard to differentiate analytically.

A more flexible sampling-based MPC, named Model Predictive Path Integral (MPPI), was proposed [32–35] to optimize for non-convex, non-linear dynamics, and high-dimensional systems. MPPI [32] was originally based on a stochastic optimal control framework and was solved by path integral control theory [50]. However, this approach is only applicable to control-affine systems. In order to scale to a large class of stochastic systems, information-theoretic MPPI [34, 35] was proposed, where the update law in MPPI can be derived based on an information-theoretic framework, without making the control affine assumption. In this paper, we focus on the latter approach.

2.2.1. Information-theoretic Framework

This section will first introduce a general discrete time control scheme and an information-theoretic interpretation of optimal control, which is based on free energy and KL-Divergence. This framework will derive the update law of MPPI, which will be used in the next section

General Control Scheme

Consider a general discrete-time stochastic dynamical system:

$$\begin{aligned} x_{t+1} &= F(x_t, v_t) \\ v_t &\sim \mathcal{N}(u_t, \Sigma) \end{aligned} \tag{2.19}$$

where $x_t \in \mathbb{R}^n$ represents state vector, $u_t \in \mathbb{R}^m$ represents the commanded control input, $v_t \in \mathbb{R}^m$ represents the actual input after applying the commanded input and F denotes the nonlinear state-transition function of the system. A sequence of inputs can be defined over a time horizon T:

$$\begin{aligned} V &= (v_0, v_1, \ldots, v_{T-1}) \\ U &= (u_0, u_1, \ldots, u_{T-1}) \end{aligned}$$

Then the base distribution $\mathbb{P}$ and controlled distribution $\mathbb{Q}_{U,\Sigma}$ can be constructed. The density function for $\mathbb{P}$ is denoted as $P(V)$ and takes the form:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2} v_t^\top \Sigma^{-1} v_t\right) \tag{2.20}$$

The density function for $\mathbb{Q}_{U,\Sigma}$ is denoted as $q(V|U,\Sigma)$ and takes the form:

$$q(V|U,\Sigma) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m|\Sigma|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(v_t - u_t)^\top \Sigma^{-1}(v_t - u_t)\right) \tag{2.21}$$

Given an initial condition x_0 and an input sequence V, the corresponding system trajectory can be obtained by recursively applying the state-transition function F. Thus a mapping from inputs V to trajectories τ can be formulated:

$$G_{x_0} : \Omega_V \to \Omega_\tau$$

where $\Omega_\tau \subset \mathbb{R}^n \times \{0, \ldots, T-1\}$ denotes the space of all possible trajectories. Then consider a state-dependent cost function for trajectories:

$$C(x_1, x_2, \ldots, x_T) = \phi(x_T) + \sum_{t=1}^{T-1} q(x_t) \tag{2.22}$$

where ϕ is a terminal cost and q is an intermediate state cost. Thus a cost function over input sequences can be defined as $S(V) : \Omega_V \to \mathbb{R}^+$:

$$S(V, x_0) = C \circ G_{x_0} \tag{2.23}$$

which will be simplified to $S(V)$ for convenience if it is not ambiguous about the initial condition.

Free energy and Relative Entropy Inequalities

In contrast to the description of free energy in section 2.1, the free energy in this chapter refers to a mathematical quantity of a control system, which originates from and takes the same form of the thermodynamic quantity – Helmholtz free energy.

Definition Given a random variable V that can denote a trajectory starting at initial condition x_0, a cost-to-go function $S(V)$ of the trajectory, probability measure $\mathbb{P}$ over V, and a positive scalar $\lambda \in \mathbb{R}^+$ called inverse temperature, the free energy of a control system is defined as:

$$F(S, \mathbb{P}, x_0, \lambda) = -\lambda \log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\right]\right) \tag{2.24}$$

It should be noticed that the scalar λ and cost-to-go function $S(V)$ are both positive, and thus the term $-\frac{1}{\lambda}S(V)$ is negative, and thus the term $\exp\left(-\frac{1}{\lambda}S(V)\right)$ is smaller than one, so the free energy is always positive.

Another quantity that is worth mentioning to describe the difference between two controlled systems is the relative entropy or called KL-Divergence.

Definition Given two probability distributions $\mathbb{P}$ and $\mathbb{Q}$, with density functions $p(V)$ and $q(V)$. Suppose that the two probability distributions are absolutely continuous with each other, which means if one density is zero, so is the other. Then the relative entropy of $\mathbb{Q}$ with respect to $\mathbb{P}$ is defined as

$$KL(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}}\left[\log\left(\frac{q(V)}{p(V)}\right)\right] \tag{2.25}$$

It should be noticed that though the relative entropy measures the difference between probability distributions, it is not a distance metric due to its lacking of symmetry, which means $KL(\mathbb{Q}||\mathbb{P})$ might be different from $KL(\mathbb{P}||\mathbb{Q})$.

With the definitions of free energy and relative entropy, a lower bound on the cost-to-go of a stochastic optimal control problem can be derived, shown by the following theorem.

Theorem 3.1 Given two probability measures $\mathbb{Q}_{U,\Sigma}$ and $\mathbb{P}$, and suppose that they are absolutely continuous with each other, then the following inequality holds:

$$F(S, \mathbb{P}, x_0, \lambda) \leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) \tag{2.26}$$

Proof. Start from the free energy of a control systemEquation 2.24. After introducing a second measure $\mathbb{Q}_{U,\Sigma}$, it can be rewritten with respect to $\mathbb{Q}_{U,\Sigma}$:

$$\begin{aligned} F &= -\lambda \log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\right]\right) \\ &= -\lambda \log\left(\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\right]\right) \end{aligned} \tag{2.27}$$

The original measure $\mathbb{P}$ can be understood as some natural base measure for trajectories, such as the probability of a trajectory under uncontrolled system dynamics. The newly introduced measure $\mathbb{Q}_{U,\Sigma}$ can be thus understood as a controlled distribution, which will be influenced by the system actuation.

Then applying Jensen's inequality yields:

$$\begin{aligned} F &= -\lambda \log\left(\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\right]\right) \\ &\leq -\lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\ &= -\lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[-\frac{1}{\lambda}S(V) + \log\left(\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] - \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{q(V|U,\Sigma)}{p(V)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) \end{aligned} \tag{2.28}$$

which reaches the end of the proof. □

The cost-to-go term $\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right]$ is under the measure $\mathbb{Q}_{U,\Sigma}$, which can be understood as a controlled distribution. The KL-Divergence between controlled measure $\mathbb{Q}_{U,\Sigma}$ and base measure $\mathbb{P}$ acts as a controlled cost penalizing deviations from the base distribution.

The Connection Between Free Energy and Optimal Control Problem

To figure out the connection between the free energy and the optimal control problem, a general setting of the discrete-time optimal control will be given first, then some simplification on the KL-Divergence between the controlled distribution $\mathbb{Q}_{U,\Sigma}$ and base distribution $\mathbb{P}$ will be done based on their density functions.

Definition Given a running cost function $\mathcal{L}(x_t, u_t)$ and a terminal cost $\phi(x_T)$, the discrete-time optimal control problem can be defined as to find the optimal control sequence within the set of admissible control sequences:

$$U^* = \underset{U\in\mathcal{U}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T) + \sum_{t=0}^{T-1}\mathcal{L}(x_t, u_t)\right] \tag{2.29}$$

where $\mathcal{U}$ is the set of admissible control sequences. It is also assumed that the running cost can be made up of an arbitrary state-dependent cost $q(x_t)$, which is similar to the intermediate state cost in Equation 2.22, and a control cost that is a quadratic function with affine term β:

$$\mathcal{L}(x_t, u_t) = q(x_t) + \frac{\lambda}{2}\left(u_t^\top \Sigma^{-1} u_t + \beta_t^\top u_t\right) \tag{2.30}$$

Now that the density functions of the base distribution $\mathbb{P}$ and controlled distribution $\mathbb{Q}_{U,\Sigma}$ are already given by Equation 2.20 and Equation 2.21, and the KL-Divergence term in Equation 2.26 can be sim-

plified as:

$$\begin{aligned} KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{q(V|U,\Sigma)}{p(V)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\prod_{t=0}^{T-1}\exp\left(-\frac{1}{2}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)+\frac{1}{2}v_t^\top\Sigma^{-1}v_t\right)\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\prod_{t=0}^{T-1}\exp\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t+\frac{1}{2}u_t^\top\Sigma^{-1}v_t+\frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t+\frac{1}{2}u_t^\top\Sigma^{-1}v_t+\frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right] \end{aligned} \tag{2.31}$$

Thus the free energy and relative entropy inequality Equation 2.26 can be reduced to:

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t+\frac{1}{2}u_t^\top\Sigma^{-1}v_t+\frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right] \end{aligned} \tag{2.32}$$

Based on the fact the covariance matrix Σ is positive-semidefinite and symmetric, it can be further derived as:

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t+\frac{1}{2}u_t^\top\Sigma^{-1}v_t+\frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right] \\ &\leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(\frac{1}{2}u_t^\top\Sigma^{-1}u_t+v_t^\top\Sigma^{-1}u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)+\sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t+2v_t^\top\Sigma^{-1}u_t\right)\right] \end{aligned} \tag{2.33}$$

Substitute Equation 2.23 into the RHS of the above inequality and denote the affine term $2v_t^\top\Sigma^{-1}$ as $\beta_t^\top$, it can be reduced to:

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)+\sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t+2v_t^\top\Sigma^{-1}u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T)+\sum_{t=1}^{T-1}q(x_t)+\sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t+\beta_t^\top u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T)+\sum_{t=0}^{T-1}\left(q(x_t)+\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t+\beta_t^\top u_t\right)\right)\right] \end{aligned} \tag{2.34}$$

where the RHS is exactly the same as the objective function in Equation 2.29. This means that the free energy provides a lower bound on the standard optimal control objective. Thus, to achieve a low-cost trajectory, instead of directly minimizing Equation 2.29, the controlled distribution $\mathbb{Q}_{U,\Sigma}$ should be pushed to make the RHS of Equation 2.26 as close as possible to the lower bound of free energy.

Optimal Distribution

Now that the inequality between the cost of a stochastic optimal problem and the free-energy term of the system has been established, it remains to show an existence of an optimal controlled distribution $\mathbb{Q}^*$ which achieves the lower bound. This section thus aims to find such optimal distribution which achieves lower cost than any other distribution via the following theorem.

Theorem 3.2 Let $\mathbb{Q}^*$ be a distribution whose density function is equal to:

$$q^*(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(V)\right) p(V) \tag{2.35}$$

where

$$\eta = \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda} S(V)\right)\right] = \exp\left(-\frac{1}{\lambda} F(S, \mathbb{P}, x_0, \lambda)\right)$$

then $\mathbb{Q}^*$ is the optimal distribution that achieves the lower bound in Theorem 3.1.

Proof. Given Equation 2.35, the KL-Divergence between the optimal distribution $\mathbb{Q}^*$ and base distribution $\mathbb{P}$ can be formulated as:

$$\begin{aligned} KL(\mathbb{Q}^*||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{q^*(V)}{P(V)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{1}{\eta}\exp\left(-\frac{1}{\lambda}S(V)\right)\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[-\frac{1}{\lambda}S(V) - \log(\eta)\right] \\ &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{Q}^*}[S(V)] - \log(\eta) \end{aligned} \tag{2.36}$$

Substituting it into the right-hand side of the inequality in Theorem 3.1 yields:

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}^*}[S(V)] + \lambda KL(\mathbb{Q}^*||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}^*}[S(V)] - \mathbb{E}_{\mathbb{Q}^*}[S(V)] - \lambda\log(\eta) \\ &= -\lambda\log(\eta) \\ &= F(S, \mathbb{P}, x_0, \lambda) \end{aligned} \tag{2.37}$$

which reaches the lower bound of the left-hand side. □

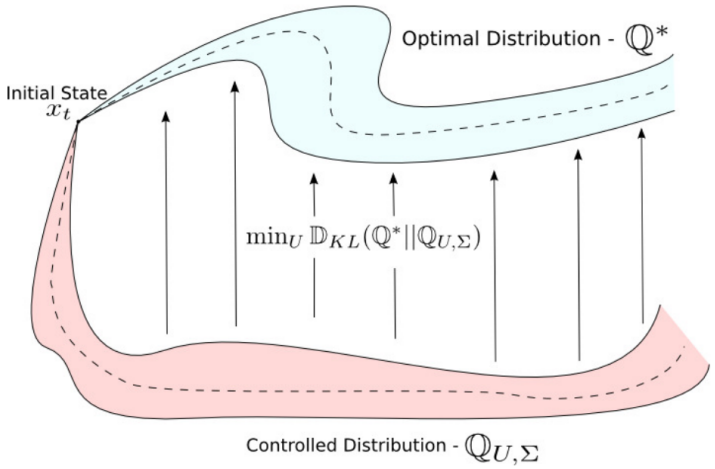


Figure 2.1: Pushing the controlled distribution to the optimal one [35]

Therefore, the controlled distribution $\mathbb{Q}_{U,\Sigma}$ should be pushed to the optimal distribution $\mathbb{Q}^*$ to achieve the lower bound of free energy, as shown in Figure 2.1. If $\mathbb{Q}_{U,\Sigma}$ is aligned with $\mathbb{Q}^*$, then sampling from $\mathbb{Q}_{U,\Sigma}$ will result in low-cost trajectories.

Push the Controlled Distribution to the Optimal Distribution

The goal of pushing the controlled distribution $\mathbb{Q}_{U,\Sigma}$ to the optimal distribution $\mathbb{Q}^*$ can be achieved by minimizing the KL-Divergence:

$$U^* = \underset{U \in \mathcal{U}}{\operatorname{argmin}}\, KL\left(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}\right) \tag{2.38}$$

where the objective KL-Divergence can be formulated as:

$$\begin{aligned} KL\left(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}\right) &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{q^*(V)}{q(V|U,\Sigma)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q^*(V)\right)\right] - \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right] \end{aligned} \tag{2.39}$$

Note that the term $\mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q^*(V)\right)\right]$ can be considered as a constant with respect to the applied control input U, thus the minimization can be reduced to:

$$\begin{aligned} U^* &= \underset{U\in\mathcal{U}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right] \\ &= \underset{U\in\mathcal{U}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbb{Q}^*}\left[-\log\left(\left((2\pi)^m|\Sigma|\right)^{\frac{1}{2}}\right) - \frac{1}{2}\sum_{t=0}^{T-1}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)\right] \\ &= \underset{U\in\mathcal{U}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}^*}\left[\frac{1}{2}\sum_{t=0}^{T-1}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)\right] \\ &= \underset{U\in\mathcal{U}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}^*}\left[\frac{1}{2}\sum_{t=0}^{T-1}\left(v_t^\top\Sigma^{-1}v_t + u_t^\top\Sigma^{-1}u_t - 2u_t^\top\Sigma^{-1}v_t\right)\right] \\ &= \underset{U\in\mathcal{U}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}^*}\left[\sum_{t=0}^{T-1}\left(\frac{1}{2}u_t^\top\Sigma^{-1}u_t - u_t^\top\Sigma^{-1}v_t\right)\right] \\ &= \underset{U\in\mathcal{U}}{\operatorname{argmin}}\left(\sum_{t=0}^{T-1}\frac{1}{2}u_t^\top\Sigma^{-1}u_t - \sum_{t=0}^{T-1}u_t^\top\Sigma^{-1}\mathbb{E}_{\mathbb{Q}^*}[v_t]\right) \end{aligned} \tag{2.40}$$

Since the objective is concave with respect to each u_t, so finding the minimum for each u_t in the unconstrained case ($\mathcal{U} = \mathbb{R}^m$) can be done by taking the gradient with respect to each u_t and setting them to zero:

$$\begin{aligned} &\Sigma^{-1}u_t^* - \Sigma^{-1}\mathbb{E}_{\mathbb{Q}^*}[v_t] = 0, \\ &\forall t \in \{0,1,\ldots,T-1\} \end{aligned}$$

Therefore, the optimal solution in the unconstrained case can be obtained:

$$\begin{aligned} &u_t^* = \mathbb{E}_{\mathbb{Q}^*}[v_t] = \int q^*(V)v_t\,\mathrm{d}V, \\ &\forall t \in \{0,1,\ldots,T-1\} \end{aligned} \tag{2.41}$$

It can be seen that the optimal open-loop control sequence is the expected actual control input sampled from the optimal distribution. However, it is still not clear how to sample from the optimal distribution, which will be covered in the next section using an approximate iterative method to compute the optimal control input.

2.2.2. Information-theoretic MPPI

After introducing the update law in an interpretation of the information-theoretic framework, this section aims to provide a complete algorithm for information-theoretic MPPI. This section will first introduce one common importance sampling technique. Subsequently, the complete information-theoretic MPPI algorithm is presented.

Iterative Importance Sampling

To estimate the optimal control solution in Equation 2.41, importance sampling technique could be used to generate the required samples [51]. Given an importance sampling control sequence U, the optimal

control solution can be rewritten as:

$$\begin{aligned} u_t^* &= \int q^*(V) v_t \, \mathrm{d}V = \int \frac{q^*(V)}{q(V|U,\Sigma)} q(V|U,\Sigma) v_t \, \mathrm{d}V \\ &= \int \omega(V) q(V|U,\Sigma) v_t \, \mathrm{d}V \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\omega(V) v_t \right] \end{aligned} \tag{2.42}$$

where the importance sampling weight term $\omega(V)$ enables to compute expectations with respect to Q^* by sampling trajectories from the system with $Q_{U,\Sigma}$. The weighting term $\omega(V)$ can be further split into two terms if the base distribution is involved:

$$\begin{aligned} \omega(V) &= \frac{q^*(V)}{p(V)} \frac{p(V)}{q(V|U,\Sigma)} \\ &= \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(V) \right) \frac{p(V)}{q(V|U,\Sigma)} \end{aligned} \tag{2.43}$$

where the first term $\frac{q^*(V)}{p(V)}$ depends on the state cost of a trajectory, and the second term $\frac{p(V)}{q(V|U,\Sigma)}$ acts like a control cost between the controlled distribution and the base distribution. Similar to the calculation in Equation 2.31, the second term can be simplified to:

$$\begin{aligned} \frac{p(V)}{q(V|U,\Sigma)} &= \frac{\exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} v_t^\top \Sigma^{-1} v_t \right)}{\exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} (v_t - u_t)^\top \Sigma^{-1} (v_t - u_t) \right)} \\ &= \exp\left(\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t - v_t^\top \Sigma^{-1} u_t \right) \right) \end{aligned} \tag{2.44}$$

Rewriting $v_t = u_t + \epsilon_t$ yields further simplification:

$$\begin{aligned} \frac{p(V)}{q(V|U,\Sigma)} &= \exp\left(\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t - (u_t + \epsilon_t)^\top \Sigma^{-1} u_t \right) \right) \\ &= \exp\left(\sum_{t=0}^{T-1} \left(-\frac{1}{2} u_t^\top \Sigma^{-1} u_t - \epsilon_t^\top \Sigma^{-1} u_t \right) \right) \\ &= \exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} \left(u_t^\top \Sigma^{-1} u_t + 2 \epsilon_t^\top \Sigma^{-1} u_t \right) \right) \end{aligned} \tag{2.45}$$

Substituting it into Equation 2.43 finally yields:

$$\omega(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \left(S(V) + \frac{\lambda}{2} \sum_{t=0}^{T-1} \left(u_t^\top \Sigma^{-1} u_t + 2 \epsilon_t^\top \Sigma^{-1} u_t \right) \right) \right) \tag{2.46}$$

$$u_t' = \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\omega(V) v_t \right], \forall t \in \{0, 1, \ldots, T-1\} \tag{2.47}$$

$$u^* = u_0' \tag{2.48}$$

$$\eta = \mathbb{E}_{\mathbb{P}} \left[\exp\left(-\frac{1}{\lambda} S(V) \right) \right] = \int \exp\left(-\frac{1}{\lambda} S(V) \right) p(V) \mathrm{d}V \tag{2.49}$$

Equation 2.46 describes the importance sampling weight between the current induced distribution and the optimal distribution. Equation 2.47 updates the importance sampling sequence. Equation 2.48 is the first element of the importance sampling sequence which can be used to approximate the stochastic optimal control problem. Equation 2.49 is the constant related to the expected cost-to-go of the trajectory with respect to the uncontrolled system dynamics, and it is also a good indicator to represent the free energy. These equations describe the *optimal information-theoretic control law* for a given distribution, which is information theoretically global optimal if the expectation can be perfectly evaluated. In practice, however, it can be locally optimal due to insufficient sampling in the state space.

Algorithm

The full algorithm of information-theoretic MPPI [35] is described below. The algorithm starts by getting the current state from a state estimator and throws K trajectory samples in parallel (from line 10 to line 23). Each trajectory is sampled by generating a random control sequence using forward dynamics, and the cost is updated for each trajectory. Once the costs for each trajectory are computed, they are transformed into probability weights (from line 24 to line 30). Finally, the control update is smoothed via a convolutional filter and the first control is sent to the actuators while the remaining sequence will be used at the next time instance (from line 31 to line 40).

Algorithm 2 Model Predictive Path Integral Control [35]

1: **Given:**
2: F, g: Dynamics and clamping function;
3: K, T: Number of samples and timesteps;
4: U: Initial control sequence;
5: Σ, ν, λ: Parameters
6: ϕ, q: Cost functions;
7: SGF: Savitsky-Galoy convolutional filter;
8: **while** task not completed **do**
9: $x \leftarrow GetStateEstimate()$;
10: /* Begin parallel sampling */
11: **for** $k = 0$ to $K - 1$ **do**
12: $\tilde{S}_k \leftarrow 0$;
13: Sample $\varepsilon^k = (\epsilon_0^k, \ldots, \epsilon_{T-1}^k), \epsilon_t^k \sim \mathcal{N}(0, \Sigma)$;
14: **for** $t = 0$ to $T - 1$ **do**
15: $v_t = u_t + \epsilon_t^k$; ▷ Equation 2.19
16: x += $F(x, g(v_t))\Delta t$;
17: // Update state cost and importance sampling weights
18: $\tilde{S}_k$ += $q(x) + \frac{\lambda}{2}\left(u_t^\top \Sigma^{-1} u_t + 2u_t^\top \Sigma^{-1} \epsilon_t\right)$; ▷ Equation 2.46
19: **end for**
20: // Add terminal cost
21: $\tilde{S}_k$ += $\phi(x)$;
22: **end for**
23: /* End parallel sampling */
24: /* Begin computing trajectory weights */
25: $\rho \leftarrow \min \tilde{S}_0, \tilde{S}_1, \ldots, \tilde{S}_{K-1}$;
26: $\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(\tilde{S}_k - \rho)\right)$; ▷ Equation 2.49
27: **for** $k = 0$ to $K - 1$ **do**
28: $\omega_k \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(\tilde{S}_k - \rho)\right)$; ▷ Equation 2.46
29: **end for**
30: /* End computing trajectory weights */
31: /* Begin control update with smoothing */
32: **for** $t = 0$ to $T - 1$ **do**
33: $U \leftarrow SGF * \left(U + \sum_{k=0}^{K-1} \omega_k \varepsilon^k\right)$;
34: **end for**
35: $SendToActuators(u_0)$;
36: **for** $t = 1$ to $T - 1$ **do**
37: $u_{t-1} \leftarrow u_t$;
38: **end for**
39: $u_{T-1} \leftarrow Initialize(u_{T-1})$;
40: /* End control update with smoothing */
41: **end while**

2.3. Discussion

Compare Active Inference with POMDP

Active inference can be formulated as a POMDP problem. The current POMDP solvers can be divided into offline solvers and online solvers.

The offline solvers will first solve the policy offline and then use it for online decision making. This suits the case when the tasks are fixed and there are no unexpected events occurring and the transition matrices or the rewards can be determined or trained beforehand, as can be shown in [52–54]. The challenges in these problems are often that the state space or action space is too large and approximations are needed. However, this approach can not deal with the newly added symbolic actions or changes in environments. Thus, when considering an easily extendable system, the active inference is more suited than offline POMDP solvers.

Online POMDP planning interleaves planning and plans execution and chooses an optimal action for the current belief only. The survey [55] lists three main ideas for online POMDP planning via belief tree search, including heuristic search, branch-and-bound pruning, and Monte Carlo sampling. The proposed algorithm *Determinized Sparse Partially Observable Tree* (DESPOT) leverages the ideas of sampling and heuristic search to approximate the standard belief tree for online planning under uncertainty. And DESPOT scales up better than other algorithms.

In this survey, much interest is put into active inference due to its flexible and unifying framework that connects to different branches of control theory, which also enables decision making with guarantees. In addition, computing probabilistic plans using active inference is cheaper and requires less memory, as shown in [56], especially in dynamic environments. Moreover, active inference allows embedding common sense ontology knowledge within discrete decision making via model parameters, where the prior preference over different plans can be used to encode habits and present the intrinsic curiosity of the agent.

Compare Active Inference with RL

Planning with active inference using expected free energy has strong connections with RL [16] since both approaches attempt to solve the optimal plan selection problem in unknown environments.

The major difference between these two approaches is the fundamental objectives, namely minimization of EFE in active inference and reward-maximization in RL [57]. To be specific, only the extrinsic term of EFE is present in RL, however, the intrinsic term is not present, which means that RL is an optimization technique based on maximizing expected reward without considering the ”intrinsic curiosity” of the agent. The essence of the intrinsic informational objective shows promising results in driving the robot to perform tasks in sparse, well-shaped, and no rewards cases [58].

In addition, active inference’s enabling goals as a prior over observations is more flexible than using rewards in RL. Since RL implicitly assumes the prior distribution is Boltzmann, the active inference is more flexible when specifying goals by using any prior distribution. Unlike the way of defining reward function in RL, which is difficult to define sometimes, active inference updates the agent’s belief until the observed outcomes match prior preferences. This is beneficial when the agents have imprecise prior preferences, and active inference enables the agents to learn prior preferences from interacting with the environment itself [57].

Compare MPPI with MPC

In order for the robot to operate in the real world safely and efficiently, the robot needs to generate an efficient trajectory to complete tasks and avoid obstacles. The most popular methods for motion planning can be categorized into sampling-based, optimization-based, and learning-based. Some of them are good alternatives but they lack some aspects for our case. For instance, MPC formulates

the planning problem as constrained optimization. However, it may become infeasible in scenarios with many constraints and it is also hard to formulate the non-convex cost functions and discontinuous dynamics. These issues leave the robot without an actionable plan. One can avoid infeasibility by relaxing constraints when necessary, but deciding which constraints to break adds additional complexity to the problem.

Sampling-based optimization methods are proposed to offer significant advantages in terms of flexibility in dealing with the constraints and cost functions. MPPI was proposed in this sampling-based optimization framework based on an information-theoretic interpretation of optimal control, and it has demonstrated its effectiveness in the domain of aggressive autonomous driving. The sampling-based nature allows the possible consideration of discontinuous dynamics and cost functions and does not require linear and quadratic approximations. This is especially beneficial in our case since we want to apply to a multi-contact and dynamic scenario. In addition, the advanced use of GPU enables fast control loop frequency in parallel sampling.

3

Scientific Paper

This chapter includes the research paper that has resulted from this thesis work, which we intend to submit to the IEEE Robotics and Automation Letters (RA-L) soon. The supporting videos can be found at https://sites.google.com/view/m3p2i-aip.

Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning

Yuezhe Zhang, Corrado Pezzato, Elia Trevisan, Chadi Salmi, Carlos Hernández Corbato, Javier Alonso-Mora

***Abstract*—Task and Motion Planning (TAMP) has progressed significantly in solving intricate manipulation tasks in recent years, but the robust execution of these plans remains less touched. Particularly, generalizing to diverse geometric scenarios is still challenging during execution. In this work, we propose a reactive TAMP method to deal with disturbances and geometric ambiguities by combining an active inference planner (AIP) for online action selection with a proposed multi-modal model predictive path integral controller (M3P2I) for low-level control. The AIP generates online alternative plans, each of which is translated into a cost function to be sampled for the proposed method. The proposed M3P2I then uses a parallelizable physics simulator for throwing different rollouts, leading to a coherent optimal solution by averaging the weighted samples based on their costs. Our method empowers real-time adaptation of action sequences to rectify failed plans, while also computing low-level motions to address dynamic obstacles or disturbances that could potentially invalidate the existing plan. Theoretical findings are validated in simulation and in the real world.**

***Index Terms*—Reactive Task and Motion Planning, Model Predictive Path Integral Control, Active Inference**

I. Introduction

Task and motion planning is a powerful class of methods for solving complex long-term manipulation problems where logic and geometric variables are influencing each other. TAMP [1] has been successfully applied in many domains such as table rearrangement, stacking blocks, or solving the Hanoi puzzle. Despite impressive recent results [2], the environments in which TAMP plans are executed are usually not dynamic and the plan is executed in open-loop [3]. Recent works [4]–[6] recognized the importance of robustifying the execution of TAMP plans in order to be able to reliably carry them out in the real world. However, these works either rely only on the adaptation of the action sequence in a plan [6]–[9] or only on the motion planning problem in a dynamic environment given a fixed plan [4], [5]. This paper thus aims to achieve reactive execution from both high-level actions and low-level motions.

A key challenge in reactive TAMP is the handling of geometric constraints that might not be known at planning time. For instance, moving a block to a desired location may necessitate pulling rather than pushing if the block is situated in a corner. This is challenging since the outcome of pushing and pulling actions can be hard to predict accurately even assuming full knowledge of the scene at planning time. Planning a sequence of push-pull actions a priori and then executing it is also prone to fail. Another scenario involves the classic pick-and-place task with sequential actions: reaching the cube, closing the gripper, lifting the cube to a pre-grasp position, and releasing the cube. However, distinct grasping poses are necessary for diverse locations, it is challenging to abstract a generalizable and robust grasping pose constraint capable of accommodating various locations while considering the fact that dynamic obstacles or human disturbances might invalidate initially planned grasping poses.

We address these challenges by proposing a control scheme that jointly addresses reactive action selection and robust low-level motion planning during execution. This paper builds upon two of our recent works: 1) active inference, a reactive action planner [7] that plans symbolic actions, and 2) a sampling-based model predictive controller [10]. In this paper, we explore the idea of connecting active inference and a sampling-based MPC by planning cost functions instead of symbolic actions. We extend the previous AIP to plan possible alternative plans to achieve the current goal. Then, we propose a Multi-Modal Model Predictive Path Integral controller (M3P2I) that samples in parallel these alternatives to blend different robot capabilities.

A. Related work

We now report relevant works with a focus on reactive TAMP and Sampling-based MPC.

a) Reactive TAMP: In [4], the authors provide a reactive MPC strategy to execute a TAMP plan as a given linear sequence of constraints. Instead of composing primitive skills, [4] derive the control law from a composition of constraints for MPC. The reactive nature of the approach allows coping with disturbances and dynamic collision avoidance during the execution of a TAMP plan. The work in [5] formulates a TAMP plan in object-centric Cartesian coordinates, showing how this allows coping with perturbations such as moving a target location. Both [4], [5], however, do not consider adaptation at the symbolic action level if a perturbation invalidates the current plan.

On the other hand, a number of papers focused on adapting and repairing high-level action sequences during execution. Authors in [11] propose to represent robot task plans as robust logical-dynamical systems. The method can effectively adapt the logic plan in order to deal with external human disturbances. Similarly, [12] presented a method to coordinate

* This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers andor sponsors.

The authors are with the Cognitive Robotics Department, TU Delft, 2628 CD Delft, The Netherlands y.zhang-130@student.tudelft.nl, {c.pezzato, e.trevisan, c.salmi, c.h.corbato, j.alonsomora }@tudelft.nl

control chains and achieved robust plan execution through plan switching and controller selection. [9] proposed instead to blend task and action planners by dynamically expanding a behavior tree at runtime through back-chaining. In [6], behavior trees and linear temporal logic have been combined into a reactive TAMP method against a cooperative or adversarial human operator which might invalidate the current plan. A recent work [13] proposes to use Monte Carlo Tree Search in combination with Isaac Gym to speed up task planning for multi-step object retrieval from clutter, where complex physical interaction is required. This is a promising direction, but [13] only focuses on high-level reasoning and executes pre-defined motions in open-loop. In [7], active inference and behavior tree were combined to provide reactive action selection in long-term tasks in partially observable and dynamic environments, which makes it particularly appealing for the problem of reactive TAMP at hand. In this paper, we extend [7] via bridging the gap to low-level reactive control by planning cost functions instead of symbolic actions.

b) Sampling-based MPC: To robustly operate in dynamic and uncertain environments, the robot must react to new situations with intelligent planning and execution. Model Predictive Control (MPC) addresses this problem via constrained optimization in a receding horizon way and has been widely used on real robotic systems [14]–[17]. However, most MPC methods rely on convexification of the constraints and cost functions, which is inflexible for high-dimensional systems such as mobile manipulation.

Sampling-based MPCs, such as MPPI [18] and STORM [19], offer promising alternatives. These algorithms make no restrictions on the convexity, non-linearity, discontinuity of the dynamics, and costs. In [20], the authors propose ensemble MPPI, a variation capable of handling complex tasks and adapting to parameter uncertainty. However, the aforementioned works have been applied for single-skill execution, such as pushing or reaching a target point, and never in the context of a longer task that requires sequential decision-making.

As pointed out in [21], one could use a high-level agent to set the cost function of the predictive controller for long-horizon cognitive tasks. We follow this line of thought and propose a method to perform the composition of cost functions for long-horizon tasks in a reactive fashion.

B. Contributions

The contributions of this work include:

- We present a method for reactive TAMP that uses active inference to plan cost functions to be minimized by MPPI instead of symbolic actions. This enables reactive execution from both high-level actions and low-level motions since the cost functions can capture both the task objectives and constraints.
- We propose a Multi-Modal MPPI (M3P2I) that is capable of sampling different alternatives to achieve a given goal and evaluating them against different costs. This enables a coherent optimal solution considering potential plans instead of relying on complex heuristics to switch between these plans.

II. Methodology

The working principle of the proposed method is depicted in Figure 1. After a more general overview, in the following, we discuss the three main parts of the scheme: *action planner*, *motion planner*, and *plan interface*.

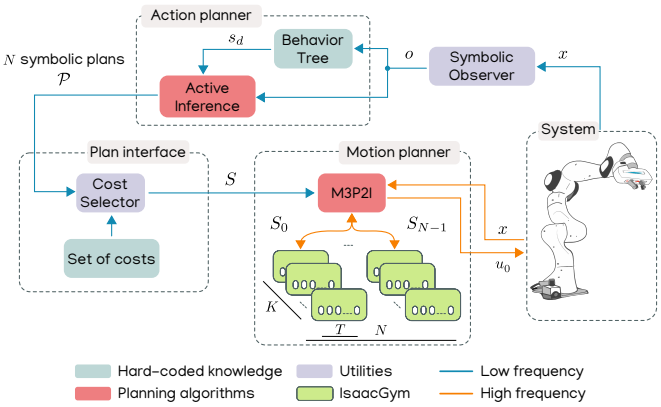


Fig. 1. Proposed control scheme. The behavior tree encodes the skeleton solution of a task. The BT sets the current desired state for Active Inference. The latter computes N alternative symbolic plans based on the current symbolic state. The plan interface links each action a_i in a plan to a cost S_i. These are sent to M3P2I that samples the control inputs using IsaacGym, and computes an action u_0. All processes are running continuously during execution, at different frequencies.

A. Overview

We first give an overview of the whole control scheme in Algorithm 1. At the initial phase, we will construct an MDP structure, which requires a manually defined action template. The action template is specific to the task and defines states, names of actions, preconditions of actions, and transition matrices corresponding to the actions. The code enters into the loop until the task is completed. We need symbolic observers to translate continuous states into discretized symbolic observations o. Then, we will feed the MDP structure and current observations o to the action planner to get the current alternative action plans $\mathcal{P}$. After the plans are generated by the task planner, they will be translated to the corresponding cost functions S, which will be fed into the M3P2I motion planner to execute the control commands.

Algorithm 1 Whole Control Scheme

```
/* Define MDP structure */
mdp_task = AIP.agent(ActionTemplate)
while task not completed do
    /* Get states from symbolic observers */
    o ← GetSymbolicObservation
    /* Get current action plans from active inference */
    P ← AIP.parell_act_sel(mdp_task, o)        ▷ Alg 2
    /* Translate action plan to cost function */
    S ← Interface(P)
    /* Execute motion commands */
    M3P2I.command(S)                           ▷ Alg 4
end while
```

B. Action planner

In this work, we operate in a continuous environment with the ability of sensing and acting through symbolic decision making. In the general case, the decision making problem will include multiple sets of states, observations, and actions. Each independent set of states is a factor, for a total of n_f factors. For a generic factor f_j where $j \in \mathcal{J} = \{1, ..., n_f\}$, the corresponding state factor is:

$$s^{(f_j)} = \left[s^{(f_j,1)}, s^{(f_j,2)}, ..., s^{(f_j,m^{(f_j)})}\right]^\top, \quad \mathcal{S} = \left\{s^{(f_j)} | j \in \mathcal{J}\right\} \tag{1}$$

where $m^{(f_j)}$ is the number of mutually exclusive symbolic values that a state factor can have. Each entry of $s^{(f_j)}$ is a real value between 0 and 1, and the sum of the entries is 1. This represents the current belief state.

Then, we define $x \in \mathcal{X}$ the continuous states of the world and the internal states of the robot are accessible through the symbolic observer. The role of this observer is to compute the symbolic observations based on the continuous state x, such that they can be manipulated by the discrete active inference agent. Observations o are used to build a probabilistic belief about the current state. Assuming one set of observations per state factor with $r^{(f_j)}$ possible values, it holds:

$$o^{(f_j)} = \left[o^{(f_j,1)}, o^{(f_j,2)}, ..., o^{(f_j,r^{(f_j)})}\right]^\top, \quad \mathcal{O} = \left\{o^{(f_j)} | j \in \mathcal{J}\right\} \tag{2}$$

Additionally, the robot has a set of symbolic skills to modify the corresponding state factor:

$$a_\tau \in \alpha^{(f_j)} = \left\{a^{(f_j,1)}, a^{(f_j,2)}, ..., a^{(f_j,k^{(f_j)})}\right\}, \quad \mathcal{A} = \left\{\alpha^{(f_j)} | j \in \mathcal{J}\right\} \tag{3}$$

where $k^{(f_j)}$ is the number of actions that can affect a specific state factor f_j. Each generic action $a^{(f_j,\cdot)}$ has associated a symbolic name, *parameters*, *pre-* and *postconditions*:

Action $a^{(f_j,\cdot)}$	**Preconditions**	**Postconditions**
`action_name(`*par*`)`	$\texttt{prec}_{a^{(f_j,\cdot)}}$	$\texttt{post}_{a^{(f_j,\cdot)}}$

where $\texttt{prec}_{a^{(f_j,\cdot)}}$ and $\texttt{post}_{a^{(f_j,\cdot)}}$ are *first-order logic predicates* that can be evaluated at run-time. A logical predicate is a boolean-valued function $\mathcal{B} : \mathcal{X} \to \{\texttt{true}, \texttt{false}\}$.

Finally, we define the logical state $l^{(f_j)}$ as a one-hot encoding of $s^{(f_j)}$. We indicate as $\mathcal{L}_c(\tau) = \left\{l^{(f_j)} | j \in \mathcal{J}\right\}$ the (time varying) *current* logical state of the world. Defining a logic state based on the probabilistic belief s built with active inference, instead of directly using the observation of the states o, increases robustness against noisy sensor readings.

In contrast to our previous work [7], this work uses active inference to generate plan alternatives. This allows the robot to perform a variety of possible plans at the same time. For example, we want the robot to push or pull an object to a goal location. But there is no prior knowledge of when to execute push or pull. If the object happens to be in the corner, the action push may fail, and the pull may work. Generating plan alternatives can be integrated with a parallel motion planner so that different actions can be evaluated in parallel in real-time. This decreases the necessity of encoding complex heuristics to reveal the spatial relationship between the objects and the environments.

Algorithm 2 Generate alternative plans using active inference

```
/* Get current action plans from active inference */
a ← Algorithm from [7]
/* Create a list to store parallel plans */
Set P ← ∅
/* Parallelize current applicable actions */
while a ∉ P or P == ∅ do
    P.append(a)
    a ← Algorithm from [7]
end while
Return P
```

The pseudocode of using active inference to generate alternative plans is shown in Algorithm 2. After the first action is found and constructed in the list $\mathcal{P}$, the algorithm will continue to search for other possible actions, if they are not extant in the current list, they will be added to the list. The algorithm will cease when no new actions are found.

C. Motion planner - Multi-Modal Model Predictive Path Integrate Control (M3P2I)

In order to solve the issue of geometric ambiguities, we propose a Multi-Modal MPPI that introduces different control and weight distributions when integrating a set of alternative plans and associated different cost functions. Assume we consider N alternative plans, each of which has K rollouts, and the cost function of plan $i, i \in [0, N)$ can be formulated as:

$$S_i(V_k) = \gamma^{T-1}\phi_i(x_{T-1,k}, v_{T-1,k}) + \sum_{t=0}^{T-2} \gamma^t q_i(x_{t,k}, v_{t,k}), \forall k \in \kappa(i) \tag{4}$$

where $V_k = [v_{0,k}, v_{1,k}, \cdots, v_{T-1,k}]$ defines the control inputs for trajectory k over a time horizon T. State $x_{t,k}$ and control input $v_{t,k}$ are indexed based on the time t and trajectory k. ϕ_i and q_i are terminal costs and intermediate state costs for plan i. The discount factor $\gamma \in [0, 1]$ is to calculate the discounted cost in the future. $\kappa(i)$ is the integer set of ranging from $i \cdot K$ to $(i + 1) \cdot K - 1$

For each cost function $S_i(V_k)$, its associated weights are calculated as ω_i:

$$\omega_i(V_k) = \frac{1}{\eta_i} \exp\left(-\frac{1}{\lambda_i}\left(S_i(V_k) - \beta_i\right)\right), \forall k \in \kappa(i) \tag{5}$$

$$\eta_i = \sum_{k \in \kappa(i)} \exp\left(-\frac{1}{\lambda_i}\left(S_i(V_k) - \beta_i\right)\right) \tag{6}$$

$$\beta_i = \min_{k \in \kappa(i)} S_i(V_k) \tag{7}$$

We use the insight in [10] to automatically tune the inverse temperature λ_i to maintain the normalization factor η_i within certain bounds. This is useful since η_i indicates the number

of samples assigned significant weights. If η_i is close to the number of samples K, an unweighted average of sampled trajectories will be taken. If η_i is close to 1 then the best trajectory sample will be taken. We observe that setting $10 < \eta_i < 30$ generates smooth trajectories for the manipulators and setting $3 < \eta_i < 10$ generates smooth trajectories for the mobile robot.

Algorithm 3 Update inverse temperature λ_i

Given:
i: index of alternative plan;
η_l, η_u: the lower bound and upper bound for η.
while $\eta_i \notin [\eta_l, \eta_u]$ **do**
  $\beta_i \leftarrow \min_{k\in\kappa(i)} S_k$; ▷ Equation 7
  $\eta_i \leftarrow \sum_{k\in\kappa(i)} \exp\Big(-\frac{1}{\lambda_i}(S_k - \beta_i)\Big)$; ▷ Equation 6
  if $\eta_i > \eta_u$ **then**
    $\lambda_i = 0.9 * \lambda_i$
  else if $\eta_i < \eta_l$ **then**
    $\lambda_i = 1.2 * \lambda_i$
  end if
end while
Return $\beta_i, \eta_i, \lambda_i$

We use μ_i to denote a sequence of mean actions of plan i over a time horizon $\mu_i = [\mu_{i,0}, \mu_{i,1}, \cdots, \mu_{i,T-1}]$, leading to the following expression:

$$\mu_i = \sum_{k\in\kappa(i)} \omega_i(V_k) \cdot V_k \tag{8}$$

At every iteration for plan i, we will sample *Halton splines* [19] from the computed mean actions μ_i to update the control sequences and feed them to the IsaacGym forward dynamics function.

In order to fuse the different motions, we also need the weights and mean for the whole control sequence. We concatenate the N cost functions $S_i(V_k), i \in [0, N)$ and simply represent it as $\tilde{S}(V_k)$. We also use κ to denote the integer set of ranging from 0 to $N \cdot K - 1$. Therefore, the weights for the whole control sequence can be calculated as:

$$\tilde{\omega}(V_k) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(\tilde{S}(V_k) - \beta\right)\right), \forall k \in \kappa \tag{9}$$

$$\eta = \sum_{k\in\kappa} \exp\left(-\frac{1}{\lambda}\left(\tilde{S}(V_k) - \beta\right)\right) \tag{10}$$

$$\beta = \min_{k\in\kappa} \tilde{S}(V_k) \tag{11}$$

The overall mean action over time horizon T is denoted as $u = [u_0, u_1, \cdots, u_{T-1}]$. The mean action at current timestep t is calculated as:

$$u_t = (1 - \alpha_u)u_{t-1} + \alpha_u \sum_{k\in\kappa} \tilde{\omega}(V_k) \cdot v_{t,k} \tag{12}$$

where α_u is the step-size that regularizes the current solution to be close to the previous one u_{t-1}.

Algorithm 4 Multi-Modal Model Predictive Path Integral Control (M3P2I)

1: **Parameters:** N, K, T;
2: **Initialize:** $\mu_i = 0, u = 0, \forall i \in [0, N)$
3: **while** task not completed **do**
4: $x \leftarrow GetStateEstimate()$;
5: /* Begin parallel sampling */
6: **for** $i = 0$ to $N - 1$ **do**
7: **for** $k \in \kappa(i)$ **do**
8: $S_k \leftarrow 0$;
9: Sample $\varepsilon^k \leftarrow SampleHaltonSplines()$;
10: $\mu_i \leftarrow Shift(\mu_i)$;
11: **for** $t = 0$ to $T - 1$ **do**
12: $x \leftarrow IsaacGymForwardSimulate(\mu_i + \varepsilon^k)$
13: $S_k \leftarrow UpdateStateCost(x)$; ▷ Eq 4
14: **end for**
15: **end for**
16: **end for**
17: /* Begin computing trajectory weights */
18: **for** $i = 0$ to $N - 1$ **do**
19: $\beta_i, \eta_i, \lambda_i \leftarrow UpdateInvTemp(i)$; ▷ Alg 3
20: $\omega_i(k) \leftarrow \frac{1}{\eta_i} \exp\Big(-\frac{1}{\lambda_i}(S_k - \beta_i)\Big), \forall k \in \kappa(i)$; ▷ Eq 5
21: $\mu_i = \sum_{k\in\kappa(i)} \omega_i(V_k) \cdot V_k$ ▷ Eq 8
22: **end for**
23: /* Begin control update */
24: $\tilde{\omega}_k = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(\tilde{S}_k - \beta\right)\right), \forall k \in \kappa$ ▷ Eq 9
25: **for** $t = 0$ to $T - 1$ **do**
26: $u_t = (1 - \alpha_u)u_{t-1} + \alpha_u \sum_{k\in\kappa} \tilde{\omega}_k \cdot v_{t,k}$ ▷ Eq 12
27: **end for**
28: $ExecuteCommand(u_0)$
29: $Shift(u)$
30: **end while**

The pseudocode for this method can be summarized in Algorithm 4. After the initialization and getting the current states, the algorithm starts parallelizing over alternative plans (Line 6), control sequences (Line 7), and time horizon (Line 11). We sample halton splines from the current mean distribution and forward them to IsaacGym to compute the costs. The costs will then be used to update the weights for each plan and compute the means (from Line 18 to Line 22). Later the overall mean action sequence will be computed (Line 26) and the first action from the mean will be executed (Line 28).

D. Plan interface

Within this block, a set of cost functions and action templates of a robot, determine the number of tasks the robot itself can perform. This block contains every cost function for a specific robot, which captures the task objectives and constraints of each action. This relationship can be stored in a knowledge base as a set of rules, such that given an action, the knowledge base can be queried to get the set of cost functions that can possibly achieve the behaviors.

III. EXPERIMENTS

This experimental section aims at highlighting the performance and potential of the scheme proposed in this paper. In

particular, we provide two different scenarios.

First, we showcase the working principle in a simplified simulation where an omnidirectional mobile robot can navigate the environment and push or pull objects in order to complete a task. The task is also transferred to a non-holonomic base, with minimal effort since there is no modeling of contacts involved. We also show the completion of tasks around dynamic obstacles as well.

Second, we showcase the application on a 7-DOF real panda arm. In addition to the normal pick-place task, we want to highlight the reactiveness of the system, including re-grasping and manipulation with human disturbances. We also show that the robot can repair plans by repeating actions or compensating for non-planned things like obstacles on a goal. We compare the performance of our methods with the cube-stacking task in [22].

We first introduce the implementation details for the two working scenarios, push-pull and pick-place. For each scenario, we report the defined action template, symbolic observers, and cost functions. We then show the simulation results for the two scenarios and the real-world experiments.

A. Implementation details

1) Push and pull:

Action template

The action template defines one state factor s^{loc}, one symbolic observation o^{loc}, and three symbolic actions $moveTo(goal), push(obj, goal), pull(obj, goal)$. The precondition to execute a push or pull action is that the robot is close to the object. We do not add complex heuristics to encode the geometric relations in the task planner to determine when to push or pull, because this is cumbersome and thus we build the contact model in the motion planner.

Action	Preconditions	Postconditions
moveTo(obj)	not closeTo(obj)	$l^{(obj)} = [1 \quad 0]^\top$
push(obj, goal)	closeTo(obj)	$l^{(goal)} = [1 \quad 0]^\top$
pull(obj, goal)	closeTo(obj)	$l^{(goal)} = [1 \quad 0]^\top$

Symbolic observer

We need one symbolic observer to estimate whether the robot R is getting close to the object O. It is defined as:

$$o^{loc} = \begin{cases} 0, |\vec{x}_R - \vec{x}_O| \leq \delta \\ 1, |\vec{x}_R - \vec{x}_O| > \delta \end{cases} \tag{13}$$

where $\vec{x}_R, \vec{x}_O$ represent the positions of the robot and the object. δ is related to the size of the robot and the object.

Cost functions

At the motion planning level, the cost function of making robot R move close to the object O can be formulated as:

$$S_{move}(R, O) = S_{dist}(R, O) = \omega_{dist} \cdot |\vec{x}_R - \vec{x}_O| \tag{14}$$

The cost function of making the robot R push object O to the goal G can be formulated as:

$$\begin{aligned} S_{push}(R, O, G) = S_{dist}(R, O) + S_{dist}(O, G) + S_{ori}(O, G) \\ + S_{align\_push}(R, O, G) + S_{dyn\_obs}(R, D) \end{aligned} \tag{15}$$

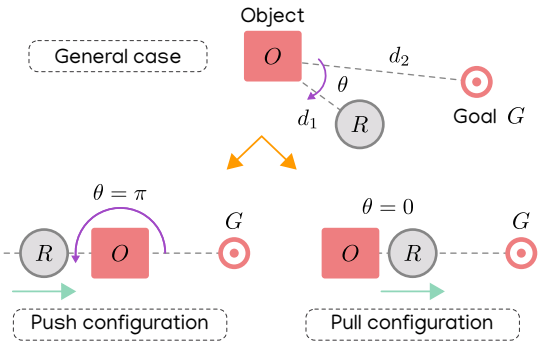


Fig. 2. Push and pull ideal configurations. The robot R has to push or pull the object O to the goal G. d_1 is the robot-object distance, while d_2 is the object-goal distance.

where minimizing $S_{dist}(O, G)$ makes the object O close to the goal G. $S_{ori}(O, G) = \omega_{ori} \cdot \phi(\Sigma_O, \Sigma_G)$ defines the orientation cost between the object O and goal G, and ϕ is given by the metric proposed in Equation 33. The align cost $S_{align\_push}(R, O, G)$ makes the object O lie at the center of robot R and goal G so that robot can push it, which is illustrated in Figure 2. The cost is related to the angle θ and function f, which are defined as:

$$cos(\theta_{ROG}) = \frac{(\vec{x}_R - \vec{x}_O) \cdot (\vec{x}_G - \vec{x}_O)}{|\vec{x}_R - \vec{x}_O||\vec{x}_G - \vec{x}_O)|} \tag{16}$$

$$f(x) = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases} \tag{17}$$

Therefore, the align cost of push can be formulated as:

$$S_{align\_push}(R, O, G) = \omega_{align\_push} \cdot f(cos(\theta_{ROG})) \tag{18}$$

Similarly, the cost function of making the robot R pull object O to the goal G can be formulated as:

$$\begin{aligned} S_{pull}(R, O, G) = S_{dist}(R, O) + S_{dist}(O, G) + S_{ori}(O, G) \\ + S_{align\_pull}(R, O, G) + S_{act\_pull}(R, O, G) + S_{dyn\_obs}(R, D) \end{aligned} \tag{19}$$

where the align cost $S_{align\_pull}(R, O, G)$ makes the robot R lie at the center of object O and goal G, and the cost is also related to the angle θ and function f as defined above. The action cost $S_{act\_pull}(R, O, G)$ allows only sample control inputs that are in the opposite direction against the object. The two costs are defined as:

$$S_{align\_pull}(R, O, G) = \omega_{align\_pull} \cdot f(-cos(\theta_{ROG})) \tag{20}$$

$$S_{action\_pull}(R, O, G) = \omega_{action\_pull} \cdot f(\frac{(\vec{x}_O - \vec{x}_R) \cdot \vec{u}}{|\vec{x}_O - \vec{x}_R||\vec{u}|}) \tag{21}$$

Moreover, in order to model pull action in the simulation, we need to exert a certain force between the object and the robot. The force is applied when the robot gets close to the object and is implemented in the IsaacGym simulator.

In addition to avoiding the collision between the obstacle and the robot, we also demonstrate collision avoidance between the block and the obstacle. This is done by using a constant velocity model to predict the position of the dynamic obstacle D in the coming horizons and by maximizing the distance between the dynamic obstacle and the robot. It can be formulated as:

$$S_{dyn\_obs}(R, D) = \omega_{dyn\_obs} \cdot e^{-|\vec{x}_R - \vec{x}_{D_{pred}}|} \tag{22}$$

where $\vec{x}_{D_{pred}}$ is the predicted position of dynamic obstacle.

2) Pick and place:

Action template

The action template defines 4 state factors $s^{reachable}, s^{holding}, s^{preplace}, s^{success}$, 4 symbolic observations $o^{reachable}, o^{holding}, o^{preplace}, o^{success}$, and 4 symbolic actions $reach(obj), pick(obj), move2place(obj, goal), place(obj)$.

Action	Preconditions	Postconditions
reach(obj)	not success	$l^{(reachable)} = \begin{bmatrix}1 & 0\end{bmatrix}^\top$
pick(obj)	reachable(obj)	$l^{(holding)} = \begin{bmatrix}1 & 0\end{bmatrix}^\top$
move2place(obj, goal)	holding(obj)	$l^{(preplace)} = \begin{bmatrix}1 & 0\end{bmatrix}^\top$
place(obj)	atPreplace(obj)	$l^{(success)} = \begin{bmatrix}1 & 0\end{bmatrix}^\top$

Symbolic observers

We need 4 symbolic observers to estimate 4 state factors. To estimate whether the gripper is close enough to the cube, we get the observation via:

$$o^{reachable} = \begin{cases} 0, l_{reach} \leq 0.012 \\ 1, l_{reach} > 0.012 \end{cases} \tag{23}$$

where $l_{reach} = |\vec{x}_{ee} - \vec{x}_O|$ measures the distance between the end effector ee and the object O.

To estimate whether the robot is holding the cube, we need:

$$o^{holding} = \begin{cases} 0, l_{gripper} < 0.065 \text{ and } l_{gripper} > 0.058 \\ 1, l_{gripper} \geq 0.065 \text{ or } l_{gripper} \leq 0.058 \end{cases} \tag{24}$$

where $l_{gripper} = |\vec{x}_{ee\_l} - \vec{x}_{ee\_r}|$ measures the distance between the two grippers.

To estimate whether the cube reaches the pre-place location, we need:

$$o^{preplace} = \begin{cases} 0, S_{dist}(O, P) < 0.01 \text{ and } S_{ori}(O, P) < 0.01 \\ 1, S_{dist}(O, P) \geq 0.01 \text{ or } S_{ori}(O, P) \geq 0.01 \end{cases} \tag{25}$$

where $S_{dist}(O, P)$ and $S_{ori}(O, P)$ measure the distance and the orientation between the object O and the pre-place location P. They are defined the same as in Equation 15. The pre-place location is a few centimeters higher than the target cube location.

Similarly, to estimate whether the cube is successfully placed on top of the target cube, we need to check:

$$o^{success} = \begin{cases} 0, S_{dist}(O, G) < 0.01 \text{ and } S_{ori}(O, G) < 0.01 \\ 1, S_{dist}(O, G) \geq 0.01 \text{ or } S_{ori}(O, G) \geq 0.01 \end{cases} \tag{26}$$

where $S_{dist}(O, G)$ and $S_{ori}(O, G)$ measure distance and the orientation between the object O and the goal location G. It is worth mentioning that the goal location is calculated directly on top of the the target cube location.

Cost functions

At the motion planning level, the cost functions of 4 actions can be formulated as:

$$\begin{aligned} S_{reach}(ee, O, \theta) = \omega_{reach} \cdot |\vec{x}_{ee} - \vec{x}_O| \\ + \omega_{tilt} \cdot (\frac{\vec{z}_{ee} \cdot \vec{z}_O}{|\vec{z}_{ee}||\vec{z}_O|} - \theta) \end{aligned} \tag{27}$$

$$S_{gripper}(ee) = \omega_{gripper} \cdot l_{gripper} \tag{28}$$

$$S_{move2place}(O, P) = S_{dist}(O, P) + S_{ori}(O, P) \tag{29}$$

$$S_{place}(O, P) = \omega_{gripper} \cdot (1 - l_{gripper}) \tag{30}$$

where $S_{reach}(ee, O, \theta)$ makes the end effector get close to the object with a grasping tilt constraint θ. When θ is close to -1, the gripper will be perpendicular to the object, when θ is close to 0, the gripper will be parallel to the plane that the object stands.

B. Simulation on mobile robots

1) Rearranging the blocks via push and pull:

We simulate rearranging the blocks via push and pull among dynamic obstacles. We use the cost functions as developed in Equation 15 and Equation 19 for push and pull. We also demonstrate collision avoidance with the dynamic obstacle.

TABLE I
SIMULATION RESULTS OF PUSH AND PULL

Skill	# of trials	Mean(std) pos error	Mean(std) ori error	# of collisions	Mean(std) time
Push	60	0.0560 (0.0137)	0.0042 (0.0056)	**0.0500** (0.2179)	**5.3530** (3.0473)
Pull	60	0.0777 (0.0762)	0.0202 (0.0519)	**0.0167** (0.1280	**9.9746** (5.8125)

The simulation results can be found in Table I. We conduct 60 trials for push and pull separately and show the performance with respect to the position error, orientation error of the blocks, number of collisions, and completion time. Push outperforms pull with respect to position error and orientation error. The average number of collisions for push and pull is 0.05 and 0.0167, which means there are only 3 and 1 collisions in total for the overall 60 trials. The completion time for push and pull ranges from 5 to 9 seconds.

2) Hybrid motions of push and pull:

The emergence of hybrid motions of push and pull is due to the issue that the block might be located in a corner, and thus one single motion cannot solve the task entirely. And it is always hard to determine heuristics that can capture the geometric information. Therefore, it is important to achieve a smooth transition between these two motions. We model the hybrid motions of push and pull by using the proposed M3P2I and incorporating both the cost functions of push and pull.

TABLE II
SIMULATION RESULTS OF HYBRID MOTIONS OF PUSH AND PULL

Case	Skill	# of trials	Mean(std) pos error	Mean(std) ori error	Mean(std) time
One corner	Push	20	0.1061 (0.0212)	0.0198 (0.0217)	6.2058 (6.8084)
	Pull	20	0.1898 (0.0836)	0.0777 (0.1294)	25.1032 (13.7952)
	Hybrid	20	**0.1052** (0.0310)	**0.0041** (0.0045)	**3.7768** (0.8239)
Two corners	Push	20	7.2679 (3.2987)	0.0311 (0.0929)	time-out
	Pull	20	0.3065 (0.1778)	0.1925 (0.2050)	32.8838 7.9240
	Hybrid	20	**0.1375** (0.0091)	**0.0209** (0.0227)	**9.9473** (3.4591)

TABLE III
SIMULATION RESULTS OF REACTIVE PICK AND PLACE

Task	Method	Training Epochs	# of trials	Mean(std) pos error	Mean(std) ori error
Normal	Ours	0	50	0.0075 (0.0036)	**0.0027** (0.0045)
	RL	1500	50	**0.0042** (0.0019)	0.2470 (0.1740)
	RL	2000	50	0.0052 (0.0067)	0.2224 (0.1591)
Reactive	Ours	0	50	**0.0117** (0.0166)	**0.0088** (0.0330)
	RL	1500	50	0.0246 (0.0960)	0.2216 (0.1501)
	RL	2000	50	0.0348 (0.1263)	0.2508 (0.1554)

We test and compare these three motions (hybrid, push, and pull) in two cases, as shown in Table II. The first case is to move an object, which is originally not in the corner, to the corner. The pull action will mostly fail in this case, which is shown by the large completion time and high position error, because the align cost in Equation 20 will force the robot to be in the corner and makes it hard to move out of the corner. The second case is to move an object, which is originally in one corner, to another corner. The push action will fail in this case, because the walls occlude the robot from reaching the other side of the object, which violates the constraint in Equation 18.

The table shows that the hybrid motions outperform push and pull in both cases, namely a lower position error, a lower orientation error, and a shorter completion time. The screenshots of push and pull can be seen at Figure 3.

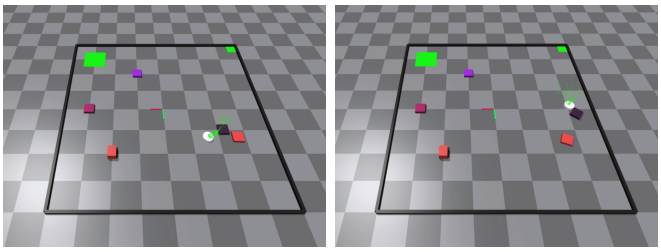


Fig. 3. Screenshots of pushing (left), pulling (right).

C. Simulation on a robot arm

1) Reactive pick and place:

In the case of a 7-DOF panda arm, we first consider the vanilla pick-and-place under disturbances. We want to pick the red cube and place it on top of the green cube. In the simulation, we model the disturbances by controlling the movements of the cubes using the keyboard during pre-grasp, placing, and after placement.

We compare the performance of our method and the off-the-shelf Actor-Critic RL method [22] in normal pick-and-place and reactive pick-and-place against disturbances. The table configuration, robot arm, and cube size in our case remain the same as in [22]. It should be noticed that the cube-stacking task in [22] only considers moving the cube on top of the other cube while neglecting the action of opening the gripper and releasing the cube. In contrast, our method exhibits fluent transitions between pick and place and also shows robustness to interferences. The results can be found in Table III, where our method is compared with the RL methods that have trained for 1500 epochs and 2000 epochs. Even though our method shows relatively inferior performance in the normal task with respect to the position error, our method outperforms the RL methods in the reactive task with respect to the average position error and orientation error.

2) Hybrid motions considering different grasping poses:

In this case, we consider grasping the object with different grasping poses. When the object is on the table, we want the end effector to pick the object on top. When the object is on the boxes, we want the end effector to pick it on its side and also avoid collision with the boxes.

To make the motion planning generalize to different object locations, we use the proposed M3P2I and incorporate the cost functions of $S_{pick}(O, G, \theta = 0)$ and $S_{pick}(O, G, \theta = 1)$ as shown in Equation 27. The only difference between these two cost functions is the different constraints for the reaching poses. The screenshots can be found at Figure 4.

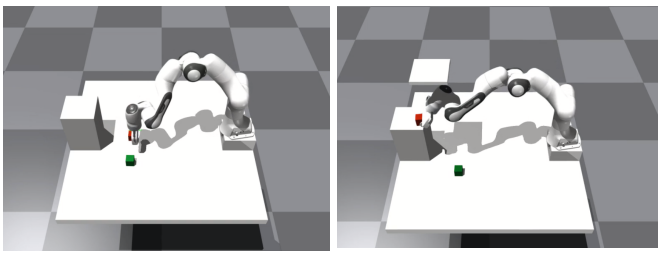


Fig. 4. Screenshots of picking an object on the table (left), and picking an object on the boxes while avoiding a moving obstacle (right).

D. Real-world experiments

We demonstrate our framework in a reactive pick-and-place scenario in real-world experiments as shown in Figure 5. The task is to pick one cube and stack it on top of the other

cube. We have shown that the robot is capable of avoiding a moving obstacle and adapting to human disturbances. The experimenter may move or rotate the cube, and steal the cube when it is in hand so that the robot will re-pick. We have also shown that M3P2I enables grasping cubes with different grasping poses, and the robot can also reach and pick the cube when it is on the boxes.

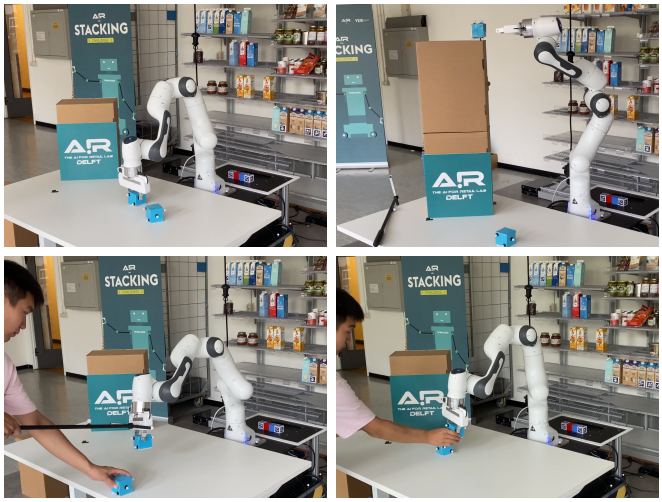


Fig. 5. Real-world experiments of picking a cube on the table (top left), picking a cube on the boxes (top right), and picking with collision avoidance and human disturbance (bottom left and right).

IV. Conclusions

In this paper, we proposed a reactive TAMP framework that combines active inference and a sampling-based MPC planner through the planning of cost functions instead of symbolic actions. In order to integrate a set of alternative plans and associated cost functions, we proposed a Multi-Modal MPPI (M3P2I) to achieve a smooth transition between these cost functions via weights updating. This novel approach allows for a more adaptable and robust control system, leading to a coherent optimal solution. We have shown that M3P2I is generalizable in combining different constraints, such as pushing and pulling for the mobile robot, and grasping objects with different grasping poses. The simulations and real-world experiments showed that the system exhibits reactiveness and robustness against human disturbances in a variety of manipulation tasks.

References

[1] 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, pp. 265–293, 2021.

[2] J. Ortiz-Haro, E. Karpas, M. Katz, and M. Toussaint, "A conflict-driven interface between symbolic planning and nonlinear constraint solving," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 518–10 525, 2022.

[3] 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.

[4] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Hönig, "Sequence-of-constraints mpc: Reactive timing-optimal control of sequential manipulation," *arXiv preprint arXiv:2203.05390*, 2022.

[5] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.

[6] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 618–12 624.

[7] C. Pezzato, C. Hernandez, S. Bonhof, and M. Wisse, "Active inference and behavior trees for reactive action planning and execution in robotics," *Transactions on Robotics (T-RO)*, 2022.

[8] N. Castaman, E. Pagello, E. Menegatti, and A. Pretto, "Receding horizon task and motion planning in changing environments," *Robotics and Autonomous Systems*, vol. 145, p. 103863, 2021.

[9] M. Colledanchise, D. Almeida, M, and P. Ögren, "Towards blended reactive planning and acting using behavior tree," *in IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[10] C. Pezzato, C. Salmi, M. Spahn, E. Trevisan, J. Alonso-Mora, and C. H. Corbato, "Sampling-based model predictive control leveraging parallelizable physics simulations," *arXiv preprint arXiv:2307.09105*, 2023.

[11] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5588–5595.

[12] J. Harris, D. Driess, and M. Toussaint, "FC^3: Feasibility-based control chain coordination," in *International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2110–2117.

[13] B. Huang, A. Boularias, and J. Yu, "Parallel monte carlo tree search with batched rigid-body simulations for speeding up long-horizon episodic robot planning," in *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[14] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[15] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*. IEEE, 2013, pp. 292–299.

[16] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "Mpc for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.

[17] M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled mobile manipulation via trajectory optimization with free space decomposition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 759–12 765.

[18] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[19] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 750–759.

[20] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2864–2871, 2020.

[21] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.

[22] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[23] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 2009.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

APPENDIX

It has been an essential issue to correctly represent 3D rotations and to efficiently evaluate the distance between them. [23] provided a detailed analysis of six metrics for measuring the distance between orientations, including Euclidean distance between the Euler angles, norm of the difference of quaternions, inner product of unit quaternions, and so on. However, these can not handle symmetric objects, which is challenging since different orientations may generate identical observations. [24] introduced ShapeMatch-Loss to focus on matching the orientations for symmetric objects. Although it does not require the specification of symmetries, it is not efficient to compute for the fully symmetric objects, such as the cubes, as we will mainly use in the experiments to interact with.

Therefore, we propose a new metric to efficiently evaluate two orientations for fully symmetric objects in a general way. We will first give the definition of general equivalence for fully symmetric objects. And then we will try to derive a proposition that meets the definition and prove it to be sufficient and necessary. Finally, we will show the proposed metric given the proved proposition.

Definition 1 We say two three-dimensional Cartesian coordinates are in general equivalence if it holds that:

$$\begin{aligned} |\vec{u}_1 \cdot \vec{v}_i| &= 1, i \in \{1,2,3\} \\ |\vec{u}_2 \cdot \vec{v}_j| &= 1, j \in \{1,2,3\} \\ |\vec{u}_3 \cdot \vec{v}_k| &= 1, k \in \{1,2,3\} \end{aligned} \tag{31}$$

where $i \neq j \neq k$, $\{\vec{u}_1, \vec{u}_2, \vec{u}_3\}$ and $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ form the orthogonal bases of the two coordinates respectively.

Proposition 1 There are two vectors (axes) in one coordinate that are collinear with some two vectors in another coordinate such that:

$$\begin{aligned} |\vec{u}_m \cdot \vec{v}_i| &= 1, m, i \in \{1,2,3\} \\ |\vec{u}_n \cdot \vec{v}_j| &= 1, n, j \in \{1,2,3\} \end{aligned} \tag{32}$$

where $m \neq n, i \neq j$. We will show that proposition 1 is equivalent to definition 1.

Theorem 1 Proposition 1 and definition 1 are equivalent.

Proof.

1) We will first show proposition 1 is a sufficient condition of definition 1. We assume $\{\vec{u}_m, \vec{u}_n, \vec{u}_p\}$ and $\{\vec{v}_i, \vec{v}_j, \vec{v}_k\}$ form the orthogonal bases of the two coordinates respectively. If there are two vectors $\vec{u}_m, \vec{u}_n$ that are collinear with $\vec{v}_i, \vec{v}_j$, then $\vec{u}_p$ is orthogonal to the plane these vectors span. So is $\vec{v}_k$. Thus $\vec{u}_p, \vec{v}_k$ are collinear. Thus it meets Equation 31 and thus meets definition 1.

2) Proposition 1 is also a necessary condition of definition 1. This is because if the two coordinates meet definition 1, there must be 4 vectors in the same plane. They are collinear. □

Theorem 1 allows us to simplify definition 1 and ignore a specific mapping between the vectors of coordinates, which would be difficult to find due to the flipping of cubes. Proposition 1 allows us to only focus on finding two pairs of collinear vectors in the two coordinates. Using this, we can derive our proposed metric to measure the difference in orientations of two symmetric objects.

Metric

$$\begin{aligned} \phi(\Sigma_u, \Sigma_v) &= \min_{i,j \in \{1,2,3\}} 1 - |\vec{u}_1 \cdot \vec{v}_i| + 1 - |\vec{u}_2 \cdot \vec{v}_j| \\ &= \min_{i,j \in \{1,2,3\}} 2 - |\vec{u}_1 \cdot \vec{v}_i| - |\vec{u}_2 \cdot \vec{v}_j| \end{aligned} \tag{33}$$

where $\Sigma_u = \{\vec{u}_1, \vec{u}_2, \vec{u}_3\}, \Sigma_v = \{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ form the orthogonal bases of two coordinates.

The major difference of our metric and ShapeMatch-Loss in [24] is the simplification of only choosing two axes in one coordinate to match with the axes in another coordinate, which is supported by theorem 1.

4

Additional Implementation Details and Experiments

This chapter provides additional information on the scientific paper. It first introduces the overall software structure of this work, and then elaborates on the implementation details and results of one additional experiment.

4.1. Software Structure

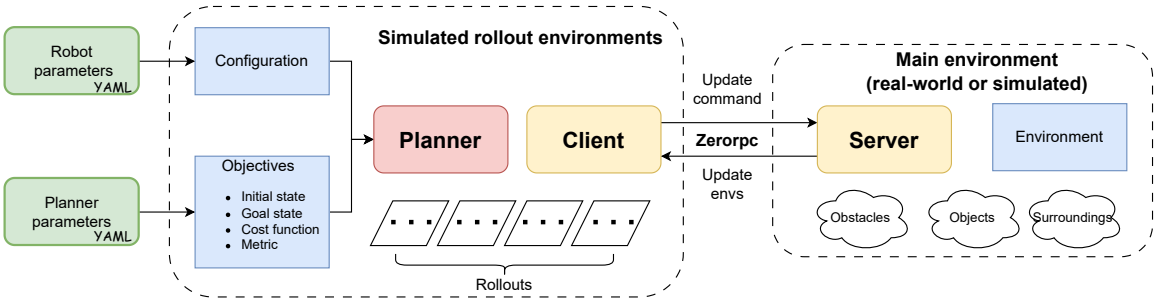


Figure 4.1: Software structure. Red and yellow components are provided. Green components should be passed before the code runs. Blue components should be implemented by the user, and they are updated at run-time.

The code structure of this work is split up into two main files as can be seen in Figure 4.1. The server file mainly reflects the setting in the main environment, which can be a real-world or simulated one, while the client file mainly simulates the planning sequences in the rollout environments. These two files communicate via the zerorpc [59], a light-weight library for distributed communication. This communication layer allows the server to update the changes in the real-world environment to the client file and allows the client to update the optimal control command to the server file to be executed in the real-world environment. This real-time planning and execution pipeline differentiates from the normal training and testing procedures in [60]. Normal RL tasks are split into the offline training phase and testing phase in IsaacGym environments separately, which are often susceptible to changes in the real dynamic world. Our work allows us to simulate real-world scenarios and populate them in the rollout environments and thus achieving planning and execution at run-time.

In addition to the communication layer, each of the other components has a clearly defined modularity:

- The PARAMETERS specifying specific configurations of the robot and planner are defined outside of the code using YAML-FILES.
- The OBJECTIVES define some important properties that a planner requires to complete the tasks, including the initial state, goal state, cost function, and metric. This component should be imple-

mented by the users. The initial state and goal state can be overridden by the YAML file and they can also be updated at run-time by the task planner.

- The PLANNER class encapsulates the reactive task and motion planning framework we introduced in chapter 3.
- The ENVIRONMENT is created by the user to run the main simulation or to reflect the real-world experiments. If it is in the physics simulator, it defines the types of obstacles, manipulated objects, and surroundings. If it is done in a real-world scenario, it should be equipped with sensors to represent the perceived world. The changes in the environment will be updated in the client file to run parallel simulations.

4.2. Navigation with Battery Requirements

We want the robot to navigate to a certain goal location while considering its battery capacity and avoiding obstacles. The battery capacity is handled by the task planner and collision avoidance is handled by the motion planner. If the battery is not enough to achieve the task, it will go to recharge at the recharging location, otherwise, it will directly go to the goal. The scenario is shown in Figure 4.2.

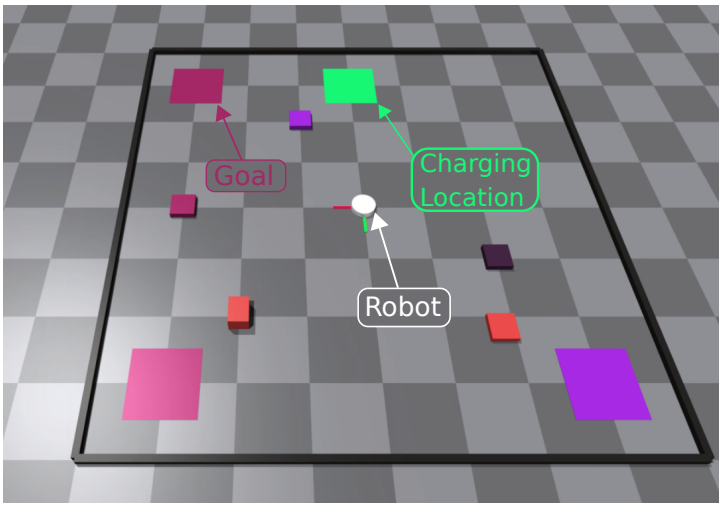


Figure 4.2: Navigation with charging scenario. The robot is an omnidirectional robot that locates at the center of the figure. The charging location is in green and the goal location is in berry.

4.2.1. Implementation

To model the navigation task with battery capacity, the complete control scheme is shown in Algorithm 3. At the initial phase, we will construct an MDP structure, which requires a manually defined action template. The action template is specific to the task and defines states, names of actions, preconditions of actions, and transition matrices corresponding to the actions. The code enters into the loop until the goal is reached. At every iteration, the battery level should be modeled and simulated. We also need to design symbolic observers to translate continuous states into discretized symbolic observations. Then, we will feed the MDP structure and current observations to the algorithm of *adaptive action selection* in [45] to get the current action plans. After the plan is generated by the task planner, it will be translated to the corresponding cost function, which will be fed into the MPPI motion planner to execute the control commands.

Algorithm 3 Navigation with battery tasks

/* *Define MDP structure* */
$mdp\_task = AIF.agent(ActionTemplate)$
$s_{batt} = 100$
while goal not reached **do**
  /* *Update battery level* */
  $s_{batt} \leftarrow SimulateBattery()$
  /* *Get states from symbolic observers* */
  $o^{loc} \leftarrow GetObsMotion(s^{loc})$
  $o^{battery} \leftarrow GetObsBattery(s^{battery})$
  $obs = [o^{loc}, o^{battery}]$
  /* *Get current action plans from active inference* */
  $\mathcal{A} \leftarrow AIF.adapt\_act\_sel(mdp\_task, obs)$ ▷ Algorithm from [45]
  /* *Translate action plan to cost function* */
  $\mathcal{S} \leftarrow Interface(\mathcal{A})$
  /* *Execute motion commands* */
  $MPPI.command(\mathcal{S})$
end while

Action Templates

The action templates define two state factors $s^{loc}, s^{battery}$, two symbolic observations $o^{loc}, o^{battery}$, and three symbolic actions $moveTo(goal)$, $goRecharge(loc), idle$. Each action is associated with its preconditions and post-conditions and they are defined in the action templates below. If the current battery is enough to complete the task, the robot will directly move to the goal location, otherwise, it will go to the charging location to recharge. The robot will recharge until the battery level is estimated to be enough to complete the task and then it will move to the goal. Finally, if the robot is close to the goal location, we will set zero velocity for the robot.

Action	**Preconditions**	**Postconditions**
moveTo(goal)	batteryEnough	$l^{(goal)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$
goRecharge(loc)	batteryNotEnough	$l^{(batteryEnough)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$
idle	at(robot, goal)	-

Simulate Battery

We simulate the battery level by assuming the fact the battery level will decline if the robot is not charging and it will increase if it is charging. It can be shown as:

$$s_{batt} = \begin{cases} s_{batt} + \alpha_{batt}, |\vec{x}_R - \vec{x}_C| \leq 0.5 \\ s_{batt} - \alpha_{batt}, |\vec{x}_R - \vec{x}_C| > 0.5 \end{cases} \tag{4.1}$$

where $\vec{x}_R$ and $\vec{x}_C$ represent the positions of the robot and the charging location. α_{batt} represents the consumption rate of the battery, the higher the value, the faster it will decline. It should also be noticed that the battery level is always bounded between 0 and 100.

Symbolic Observers

We need two symbolic observers to estimate whether the goal has been reached and to estimate whether the current battery level is sufficient to complete the task. They are defined as:

$$o^{loc} = \begin{cases} 0, |\vec{x}_R - \vec{x}_G| \leq 0.5 \\ 1, |\vec{x}_R - \vec{x}_G| > 0.5 \end{cases} \tag{4.2}$$

$$o^{battery} = \begin{cases} 0, \dfrac{|\vec{x}_R - \vec{x}_G| \cdot \alpha_{dist} \cdot \alpha_{batt}}{s_{batt} - 60} < 1 \text{ and } s_{batt} > 60 \\ 1, \dfrac{|\vec{x}_R - \vec{x}_G| \cdot \alpha_{dist} \cdot \alpha_{batt}}{s_{batt} - 60} \geq 1 \text{ or } s_{batt} \leq 60 \end{cases} \tag{4.3}$$

where $\vec{x}_G$ represents the positions of the robot and the navigation goal. s_{batt} represents the current battery level ranging from 0 to 100. α_{dist} represents the parameter related to distance efficiency, where we assume the distance the robot can run is proportional to its battery consumption, so the higher the value, the shorter distance the robot can run given a certain battery level.

Cost function
At the motion planning level, the cost function of making robot R move to the goal G can be formulated as:

$$S_{move}(R,G) = S_{dist}(R,G) + S_{ori}(R,G) + S_{coll}(R) \tag{4.4}$$

where the functions can be defined as:

$$S_{dist}(R,G) = \omega_{dist} \cdot |\vec{x}_R - \vec{x}_G| \tag{4.5}$$
$$S_{ori}(R,G) = \omega_{ori} \cdot \phi(\Sigma_R, \Sigma_G) \tag{4.6}$$
$$S_{coll}(R) = \omega_{coll} \cdot \sum_{i \in I_{non-move}} F_{contact}(R,i) \tag{4.7}$$

where $S_{dist}(R,G)$ minimizes the distance between the robot R and the goal G. $\phi(\Sigma_R, \Sigma_G)$ defines the orientation cost between the robot R and goal G, and ϕ is given by the metric proposed in the paper. The collision avoidance is modeled by $F_{contact}(R,i)$, which defines the contact force between the robot R and the non-moveable object i, and the function is provided by the IsaacGym simulator.

Similarly, the constraint of $goRechage(loc)$ can be formulated as:

$$S_{recharge}(loc) = S_{dist}(loc) + S_{ori}(loc) + S_{coll} \tag{4.8}$$

4.2.2. Experimental Results

We simulate the fusion of internal needs and external goals, including behavior shaping according to internal battery needs, and navigation among obstacles. We use the control scheme as shown in Algorithm 3 and test it in IsaacGym. The simulation results can be divided into 2 cases considering different values of battery consumption rate α_{batt}.

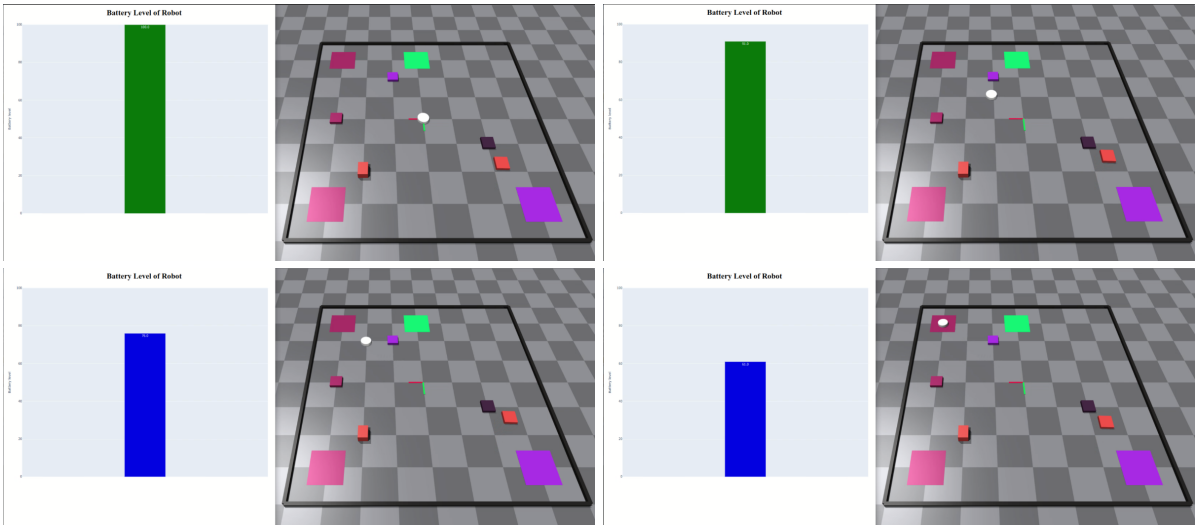


Figure 4.3: Screenshots of navigation with battery capacity in case 1. The screenshots are placed in chronological order (top left, top right, bottom left, bottom right).

In the first case, we set the battery consumption rate relatively low ($\alpha_{batt} = 0.8$). This suggests that the battery level declines at a low speed so that the battery estimator will estimate that the current battery

is enough to complete the whole task without the need to recharge. The screenshots of the simulation process for this case are shown in Figure 4.3. The screenshots are placed in chronological order (top left, top right, bottom left, bottom right). In the initial phase, the robot is at the center with a battery level of 100 as shown in the top left figure. As time goes by, figures in the top right and bottom left show that the battery level declines to 91.0 and 71.0. Now the battery estimator guesses the robot would arrive at the goal location with a battery level larger than 60, so $o^{battery}$ is set to 0. Since the robot is not close to the goal yet, o^{loc} is set to 1. These two observations will lead to active inference to select moving to the goal as the current optimal action instead of going to recharge. Therefore, in the final bottom right figure, the robot arrives at the goal location with a battery level of 61.0. This means the robot succeeds in completing the task and will be set to 0 velocities.

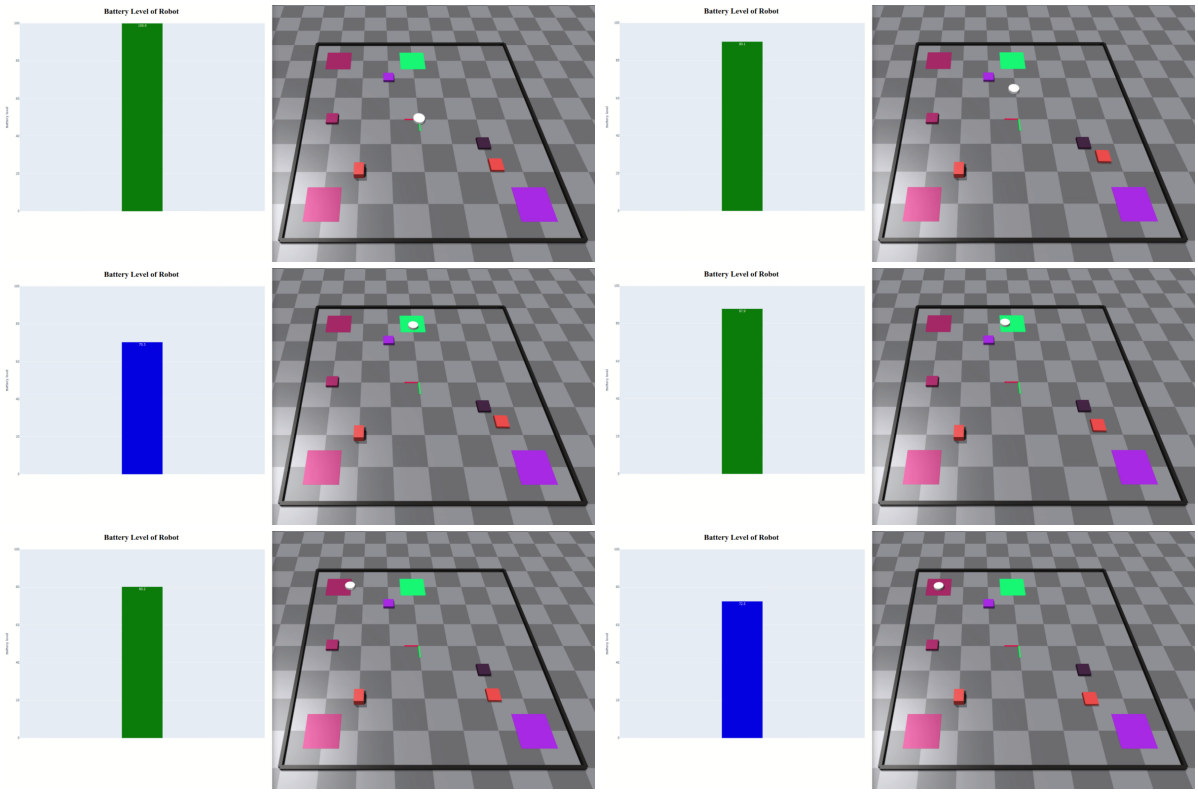


Figure 4.4: Screenshots of navigation with battery capacity in case 2. The screenshots are placed in chronological order (top left, top right, middle left, middle right, bottom left, bottom right).

In the second case, we set the battery consumption rate relatively low ($\alpha_{batt} = 1.2$), indicating that the battery level declines at a relatively high speed. The screenshots of the simulation process for this case are shown in Figure 4.4. The screenshots are placed in chronological order (top left, top right, middle left, middle right, bottom left, bottom right). In the initial phase, the robot is at the center with a battery level of 100 as shown in the top left figure. When the robot starts to move, the figure in the top right shows that the battery level declines to 90.1 and the robot is heading towards the charging location directly. This is the major difference from the first case mainly because the battery estimator guesses the robot can not arrive at the goal location with a battery level larger than 60, so $o^{battery}$ is set to 1. Since the robot is not close to the goal yet, o^{loc} is set to 1. These two observations will lead to active inference to select going to recharge as the current optimal action instead of moving to the goal. In the next time step, the robot arrives at the charging location with a battery level of 70.3 as shown in the figure middle left. Later it gets recharged until the battery is estimated to be enough to arrive at the goal location, which is shown in the middle right figure with battery 87.9. Then the robot moves towards the goal location as shown in the bottom left figure and finally arrives at the goal location with a battery level of 72.5 as shown in the bottom right figure.

5 Discussion

In this chapter, we discuss key aspects of the simulation and real-world experiments.

5.1. Sim-to-real Issues

Sim-to-real is known as transferring robot skills acquired in the physics simulator to the real world. It draws attention since it is cheaper, safer, and more informative to perform experiments in simulation than in the real world. However, uncertainties and discrepancies between simulated and real-world environments could present challenges for achieving precise movements and manipulation. In our experiments, we found these issues worth noticing.

5.1.1. Mismatching of Gripper Control Modes

In the simulation, the actions of closing and opening the grippers are controlled by setting the continuous velocity commands. When we set the velocity of one gripper to be positive, it is opening, and when we set the velocity to be negative, it is closing. Therefore, in our sampling-based MPC algorithm for the simulation, we only need to sample the control inputs of velocities ranging from -0.8 to 0.8m/s. In addition, we can control the movements of the left gripper and the right gripper separately.

However, there are two firsthand issues when we use the real 7-DOF Panda arm. First, there is only one degree-of-freedom for the two grippers. In addition, opening or closing the grippers is set by the discrete ROS actions instead of by the continuous velocity.

To solve the mismatching of the degree-of-freedom of the grippers, we only sample one control input of one gripper in the simulation and set the same velocity to the other gripper. To address the issue of sending commands in the real grippers, we use two action clients. One action client sends *MoveAction* to close the grippers, and the other action client sends *GraspAction* to open the grippers. The two actions are sent with a constant speed 0.1 and different widths. The *GraspAction* is associated with a goal of width 0.38 to open the grippers. This means opening the gripper through the GraspAction makes the finger having an aperture of 7.6 cm. The cube to grasp is roughly 6cm so that we can easily place it. The *MoveAction* is associated with a width of 0 to make the the grippers closed. This will make the two grippers fully closed if there is no cube in hand, and will make the grippers grasp the cube if it is in hand.

We further found that if we send the close commands many times and send an open command once, the gripper will open only once and close again. This is because multiple close commands are stored in a queue, when the open command is sent, the gripper will open first since it has a higher priority, but the original commands are still in the queue and are not canceled, so they will trigger the close action again. To solve the issue, we set a flag to send only one close command until an open command is sent. Similarly, we set another flag to send only one open command until a close command is sent.

5.1.2. Mismatching of Physical Properties

Another issue we encounter is the imprecise physical parameters and properties in the real world. For example, the size, mass, friction, and gravity of the cubes do matter in the pick-and-place task. We manually set these parameters in the simulation but online system identification is not performed. Ensembling rollouts with parameter variations, as demonstrated in [61], could further improve the performance.

In addition, resetting the position and velocity of the cube via OptiTrack cameras is also very essential at every iteration step. It might also be challenging when the cubes get close to each other or the cube is grasped in hand, their associated markers will result in unstable and flashing perception in the OptiTrack. To alleviate the issue, we focus on redistributing the markers and recalibrations of the motion capture system. Future work could focus on a combination of other sensors such as cameras and lidars to obtain precise geometric information.

5.2. Collision Avoidance and Computation Efficiency

The computational demands of planning and control with our method can be high when extending the time horizon to several seconds. To keep the time horizon limited for real-time control while preventing being trapped in local minima, future work should incorporate global planning techniques such as A*, RRT, and Probabilistic Roadmaps (PRM) to guide the local planner. In addition, incorporating additional sensor support such as lidars and composite signed distance fields could be highly beneficial. This would enable, for instance, collision avoidance without the need for complex simulated worlds.

6 Conclusion

This conclusive chapter summarizes the content presented in this work, and the answers to the initial research questions. Potential future directions closely tied to this work are also delineated.

6.1. Summary

In pursuit of reliable reasoning and robust execution of robot actions in dynamic environments, TAMP methods render a powerful framework for complex long-term manipulation skills. However, previous works either rely on the adaption of the action sequence in a plan or only on the motion planning problem given a fixed plan. Less attention has been paid to robustifying the execution of TAMP plans. In addition, a key challenge in reactive TAMP is the handling of geometric constraints. Since tasks can be carried out in multiple ways depending on the geometry of the problem, these geometric ambiguities remain especially when the correct set of geometric constraints is hard to determine at planning time. Section 1.1 motivated this aspect in two scenarios, namely a push-pull scenario and a pick-place scenario.

Section 1.2 presented an overview of the fields that entail the topic of the thesis, and chapter 2 provided the necessary knowledge of the two building blocks of the thesis. Active inference uses free energy to describe the properties of an agent in an environment, and by minimizing expected free energy at run time, Bayes-optimal behavior can be obtained. POMDP methods are also capable of generating reactive behaviors in uncertain, dynamic environments, but offline POMDP solvers only suit the case when the tasks are fixed and there are no unexpected events and additional actions. There are also some good alternatives of online POMDP solvers, but active inference suits our case considering its appealing aspects, namely the efficiency in continual online planning and the flexibility in encoding parameterized habits and preferences. Among a variety of motion planning methods, MPPI attracts our attention due to its flexibility in dealing with constraints and cost functions. Moreover, massive sampling allows the robot to have more opportunities to explore the environments and thus is beneficial to generate efficient contact-rich manipulation skills.

Chapter 3 encapsulated the main body of this work in the form of a scientific paper. In the paper, we proposed a reactive TAMP framework that combines active inference and MPPI through the planning of cost functions instead of symbolic actions. This enables reactive execution from both high-level actions and low-level motions since the cost functions can capture both the task objectives and constraints. Besides, to address the issue of geometric ambiguities, we proposed a Multi-Modal MPPI (M3P2I) that is capable of sampling different alternatives and evaluating them against different costs. This enables smooth transitions between these cost functions via weights updating, which reduces the number of required heuristics in the task planner and results in a coherent optimal solution. In addition, we proposed a new metric to efficiently evaluate two orientations for fully symmetric objects in a general way. This was to fix the geometric issues for fully symmetric objects, such as the cube, whose different orientations may generate identical observations. We showed that this metric is more simplified compared

with the ShapeMatch-Loss in [62].

We have demonstrated that the framework exhibits reactiveness in different scenarios, including battery charging as shown in section 4.2, push-pull among obstacles, and pick-place with disturbances as shown in 3. We showed that our framework outperforms the off-the-shelf RL method in the reactive pick-place task in terms of position error and orientation error. We have also shown that M3P2I is generalizable in combining different constraints. In the push-pull case, the robot was able to smoothly switch the actions between push and pull in order to make the object to the corner. The results showed that the hybrid motions were better than any individual action of push or pull in terms of task time efficiency, position errors, and orientation errors of the object. In the pick-place case, we demonstrated the robot was able to generalize to either reach the cube on the boxes or reach the cube on the table with the end effector perpendicular to the table. The real-world experiments showed that the system exhibits reactiveness and robustness against human disturbances in a variety of manipulation tasks.

6.2. Answers to the Research Questions

How can Task and Motion Planning (TAMP) incorporate reactive behaviors from both high-level action selection and low-level motions in multi-task, contact-rich and dynamic environments?

This work aimed at providing a solution in robustifying the execution of TAMP plans in order to reliably carry them out in the real world. We proposed a reactive TAMP framework that combines active inference and a sampling-based MPC planner through the planning of cost functions instead of symbolic actions. The cost functions can capture not only the task objectives but also the constraints, including requirements of grasping poses, smooth motions, and collision avoidance with obstacles. This scheduling of costs is flexible in both high-level decision-making and low-level motions, enabling the robot to achieve both long-term and short-term cognitive aspects of the tasks.

How to address geometric ambiguities, which are hard to determine at planning time, within a reactive TAMP framework?

The geometric ambiguities refer to the selection of correct geometric constraints at run-time as motivated in 1.1. We demonstrated these in two cases, including a hybrid push-pull task for a mobile robot and a pick-place task with different grasping poses for a manipulator. Even assuming full ethic knowledge of the scene at planning time, since the outcome of pushing and pulling actions can be hard to predict accurately, planning a sequence of push-pull actions a priori and then executing it is prone to fail. Instead, we seek a solution capable of blending at runtime different robot capabilities to achieve smooth transitions between skills without hard-coded switching. We extended previous active inference to plan possible alternative plans, which will be integrated into the motion planner as associated cost functions. Further, we proposed a Multi-Modal MPPI (M3P2I) to achieve a smooth transition between these cost functions via weights updating. This was done by updating the weight distributions of different plans at first and by incorporating different weight distributions into a cohesive distribution of actions. This novel approach allows for a more adaptable and robust control system, leading to a coherent optimal solution.

6.3. Future Work

This work followed the line of thought as pointed out in [36] to use a high-level agent to set the cost function of the predictive controller for long-horizon cognitive tasks. The cost functions are represented as a linear sum of multiple functions that encapsulate task objectives and motion constraints. It would be nice to explore more complex combinations and compositions of these building blocks of constraints or skills to engender sophisticated and intelligent behaviors. Undoubtedly, tuning suitable weights for optimal performance is tricky and time-consuming. It would be desirable to design an algorithm that can automatically tune these hyperparameters and check the performance effortlessly. [63] provides a solution to frame the parameter tuning as a constrained optimization problem. It conducts different trials under different parameter sets. After each trial is simulated, the performance is evaluated under

an objective function, which is made up of a sum of metrics, and a new parameter set is suggested based on the history of trials. Finally, the best parameter set is extracted from all trials. However, it only tests it for the trajectory generation using optimization fabrics, more tests need to be done for sampling-based MPC.

References

[1] AIR-Lab. http://aiforretail.ai/. Accessed: 2022-07-17.

[2] Caelan Reed Garrett et al. “Integrated task and motion planning”. In: *Annual review of control, robotics, and autonomous systems* 4 (2021), pp. 265–293.

[3] Joaquim Ortiz-Haro et al. “A Conflict-driven Interface between Symbolic Planning and Nonlinear Constraint Solving”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10518–10525.

[4] 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.

[5] Marc Toussaint et al. “Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation”. In: *arXiv preprint arXiv:2203.05390* (2022).

[6] Toki Migimatsu and Jeannette Bohg. “Object-centric task and motion planning in dynamic environments”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 844–851.

[7] Shen Li et al. “Reactive task and motion planning under temporal logic specifications”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 12618–12624.

[8] Corrado Pezzato et al. “Active inference and behavior trees for reactive action planning and execution in robotics”. In: *Transactions on Robotics (T-RO)* (2022).

[9] Nicola Castaman et al. “Receding Horizon Task and Motion Planning in Changing Environments”. In: *Robotics and Autonomous Systems* 145 (2021), p. 103863.

[10] M. Colledanchise et al. “Towards blended reactive planning and acting using behavior tree”. In: *in IEEE International Conference on Robotics and Automation (ICRA)* (2019).

[11] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[12] Erez Karpas and Daniele Magazzeni. “Automated planning for robotics”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 417–439.

[13] Malik Ghallab, Dana Nau, and Paolo Traverso. “The actor’s view of automated planning and acting: A position paper”. In: *Artificial Intelligence* 208 (2014), pp. 1–17.

[14] Dana Nau, Malik Ghallab, and Paolo Traverso. “Blended planning and acting: Preliminary approach, research challenges”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[15] Michele Colledanchise, Diogo Almeida, and Petter Ögren. “Towards blended reactive planning and acting using behavior trees”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8839–8845.

[16] Pablo Lanillos et al. “Active inference in robotics and artificial agents: Survey and challenges”. In: *arXiv preprint arXiv:2112.01871* (2021).

[17] Karl Friston. “The free-energy principle: a unified brain theory?” In: *Nature reviews neuroscience* 11.2 (2010), pp. 127–138.

[18] Corrado Pezzato, Riccardo Ferrari, and Carlos Hernández Corbato. “A novel adaptive controller for robot manipulators based on active inference”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2973–2980.

[19] Mohamed Baioumy et al. “Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference”. In: *2021 European Control Conference (ECC)*. IEEE. 2021, pp. 1119–1125.

[20] Corrado Pezzato et al. “Active inference for fault tolerant control of robot manipulators with sensory faults”. In: *International Workshop on Active Inference*. Springer. 2020, pp. 20–27.

[21] Corrado Pezzato et al. “Active inference and behavior trees for reactive action planning and execution in robotics”. In: *arXiv preprint arXiv:2011.09756* (2020).

[22] Tomás Lozano-Pérez and Michael A Wesley. “An algorithm for planning collision-free paths among polyhedral obstacles”. In: *Communications of the ACM* 22.10 (1979), pp. 560–570.

[23] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[24] Lydia E Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[25] Steven M LaValle et al. “Rapidly-exploring random trees: A new tool for path planning”. In: (1998).

[26] Steven M LaValle and James J Kuffner Jr. “Randomized kinodynamic planning”. In: *The international journal of robotics research* 20.5 (2001), pp. 378–400.

[27] Nathan Ratliff et al. “CHOMP: Gradient optimization techniques for efficient motion planning”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 489–494.

[28] Mrinal Kalakrishnan et al. “STOMP: Stochastic trajectory optimization for motion planning”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[29] John Schulman et al. “Motion planning with sequential convex optimization and convex collision checking”. In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[30] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. “Movement imitation with nonlinear dynamical systems in humanoid robots”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 2. IEEE. 2002, pp. 1398–1403.

[31] S Mohammad Khansari-Zadeh and Aude Billard. “Learning stable nonlinear dynamical systems with gaussian mixture models”. In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[32] Grady Williams et al. “Aggressive driving with model predictive path integral control”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1433–1440.

[33] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. “Model predictive path integral control: From theory to parallel computation”. In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[34] Grady Williams et al. “Information theoretic MPC for model-based reinforcement learning”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1714–1721.

[35] Grady Williams et al. “Information-theoretic model predictive control: Theory and applications to autonomous driving”. In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1603–1622.

[36] Taylor Howell et al. “Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo”. In: *arXiv preprint arXiv:2212.00541* (2022).

[37] Caelan Reed Garrett et al. “Integrated Task and Motion Planning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 265–293.

[38] C. Paxton et al. “Representing Robot Task Plans as Robust Logical-Dynamical Systems”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 5588–5595. DOI: 10.1109/IROS40897.2019.8967861.

[39] Jason Harris, Danny Driess, and Marc Toussaint. “FC^3: Feasibility-Based Control Chain Coordination”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 2110–2117. DOI: 10.1109/CASE49997.2022.9926658.

[40] Ziyi Zhou et al. “Reactive Task Allocation and Planning for Quadrupedal and Wheeled Robot Teaming”. In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. 2022, pp. 2110–2117. DOI: 10.1109/CASE49997.2022.9926658.

[41] Baichuan Huang, Abdeslam Boularias, and Jingjin Yu. “Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning”. In: *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022.

[42] Samuel J Gershman. “What does the free energy principle tell us about the brain?” In: *arXiv preprint arXiv:1901.07945* (2019).

[43] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. “Variational inference: A review for statisticians”. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[44] Lancelot Da Costa et al. “Active inference on discrete state-spaces: A synthesis”. In: *Journal of Mathematical Psychology* 99 (2020), p. 102447.

[45] Corrado Pezzato et al. “Active inference and behavior trees for reactive action planning and execution in robotics”. In: *IEEE Transactions on Robotics* (2023).

[46] Moses Bangura and Robert Mahony. “Real-time model predictive control for quadrotors”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 11773–11780.

[47] Tom Erez et al. “An integrated system for real-time model predictive control of humanoid robots”. In: *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*. IEEE. 2013, pp. 292–299.

[48] Nicola Scianca et al. “MPC for humanoid gait generation: Stability and feasibility”. In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1171–1188.

[49] Max Spahn, Bruno Brito, and Javier Alonso-Mora. “Coupled mobile manipulation via trajectory optimization with free space decomposition”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 12759–12765.

[50] Hilbert J Kappen. “Linear theory for control of nonlinear stochastic systems”. In: *Physical review letters* 95.20 (2005), p. 200201.

[51] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*. Vol. 1. 2. Springer, 2001.

[52] Bruno Lacerda et al. “Probabilistic planning with formal performance guarantees for mobile service robots”. In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1098–1123.

[53] Michael Painter, Bruno Lacerda, and Nick Hawes. “Convex hull Monte-Carlo tree-search”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 30. 2020, pp. 217–225.

[54] David Silver et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).

[55] Nan Ye et al. “DESPOT: Online POMDP planning with regularization”. In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 231–266.

[56] Mohamed Baioumy et al. “On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 819–829.

[57] Noor Sajid et al. “Active inference: demystified and compared”. In: *Neural computation* 33.3 (2021), pp. 674–712.

[58] Alexander Tschantz et al. “Reinforcement learning through active inference”. In: *arXiv preprint arXiv:2002.12636* (2020).

[59] Zerorpc. `https://www.zerorpc.io/`. Accessed: 2023-08-14.

[60] Viktor Makoviychuk et al. “Isaac gym: High performance gpu-based physics simulation for robot learning”. In: *arXiv preprint arXiv:2108.10470* (2021).

[61] Ian Abraham et al. “Model-based generalization under parameter uncertainty using path integral control”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2864–2871.

[62] Yu Xiang et al. “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[63] Max Spahn and Javier Alonso-Mora. “Autotuning Symbolic Optimization Fabrics for Trajectory Generation”. In: *arXiv preprint arXiv:2302.06922* (2023).

A

Additional Figures

A.1. Additional Figures for Push and Pull

In the experiments of *rearranging the blocks via push and pull* in chapter 3, the screenshots of push and pull can be shown in Figure A.1 and Figure A.2, which are placed in chronological order (from top left to top right, and from bottom left to bottom right). The purple object is the one to manipulate (push or pull), the red object is the dynamic obstacle to avoid, and the green area is the goal location. The green lines are the sampled trajectories. In the pushing scenario, the robot first tries to approach the object and pushes the object in the goal direction. Considering that the dynamic obstacle is nearby, the robot is conservative in pushing as shown in the top right figure in Figure A.1. The robot pushes the object directly to the goal location until the dynamic obstacle is moving away, as shown in the bottom right figure in Figure A.1.

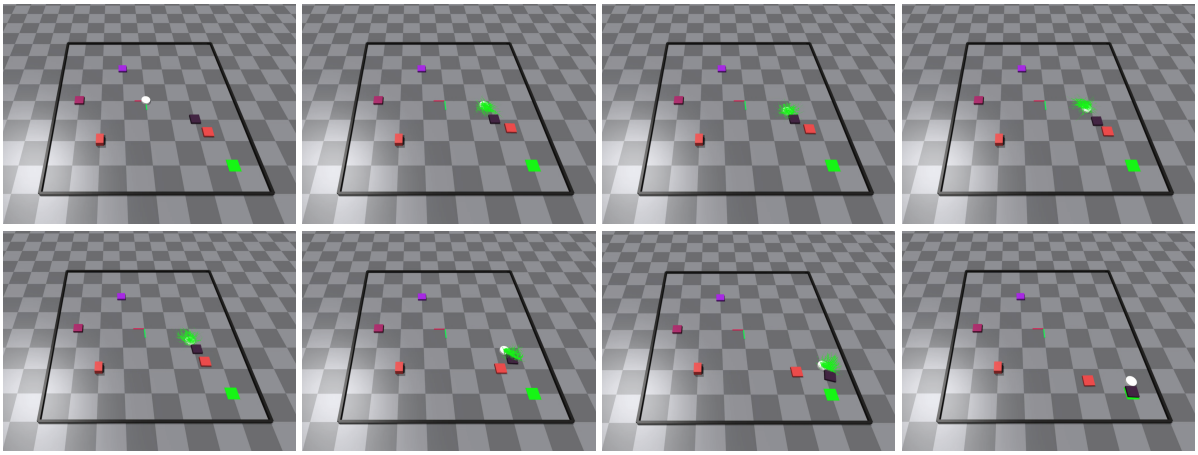


Figure A.1: Screenshots of push action.

In the pulling scenario, the robot first tries to circumvent the dynamic obstacle as shown in the top figure in Figure A.2 and then approaches the object. Then it pulls the object to the goal location as shown in Figure A.2.

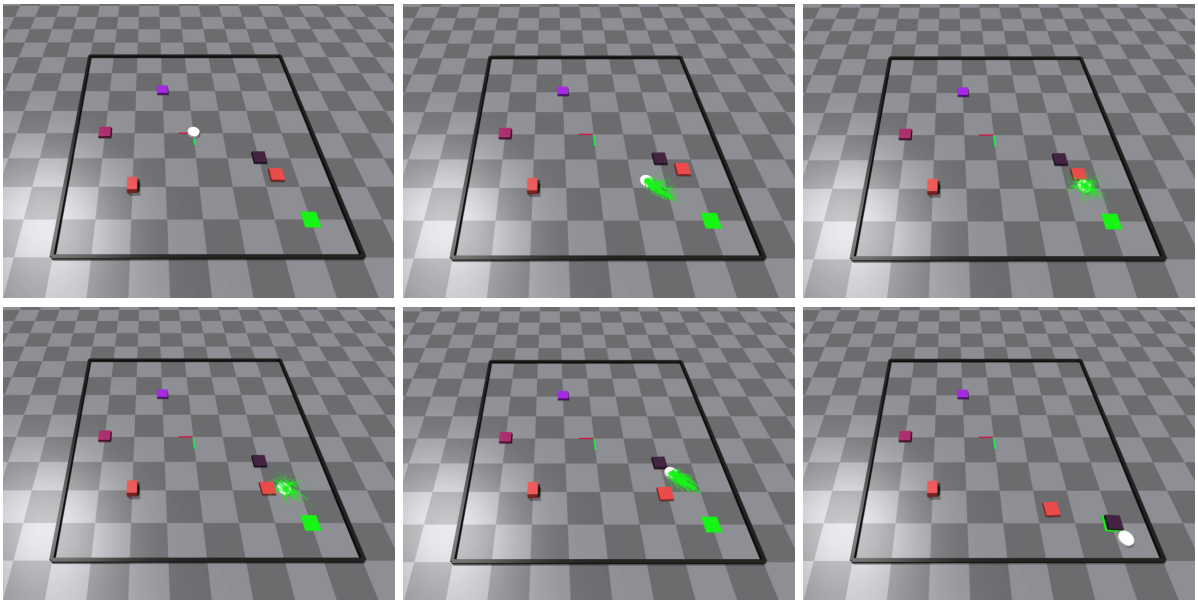


Figure A.2: Screenshots of pull action.

A.2. Additional Figures for Hybrid Push and Pull

In the experiments of *hybrid motions of push and pull* in chapter 3, the metrics that are used to depict the process of push, pull, and hybrid actions are shown in Figure A.3, Figure A.4, and Figure A.5. The first column of the metrics depicts the trajectories of the object in blue and the robot in orange. The second column of the metrics depicts the distance between the robot and the object in pink and the distance between the object and the goal in purple. The third column of the metrics depicts the cosine of the angle between the robot, object, and goal.

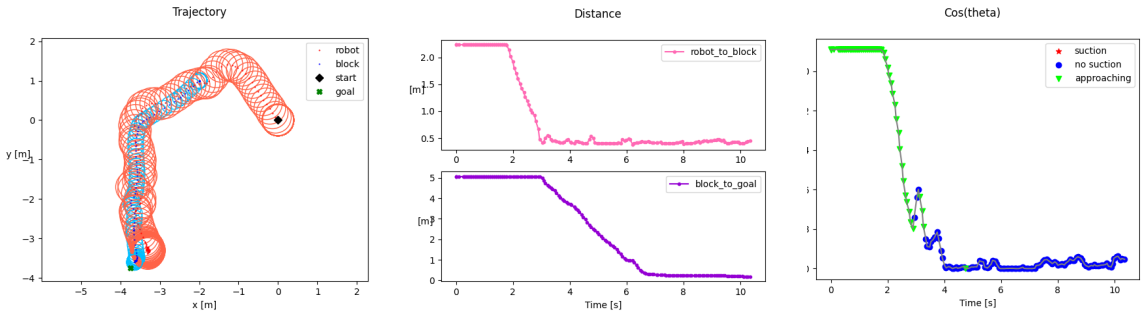


Figure A.3: Metrics of push action.

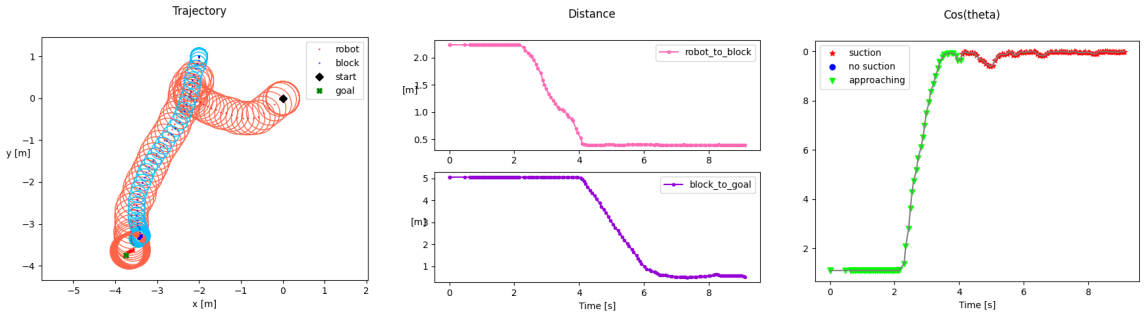


Figure A.4: Metrics of pull action.

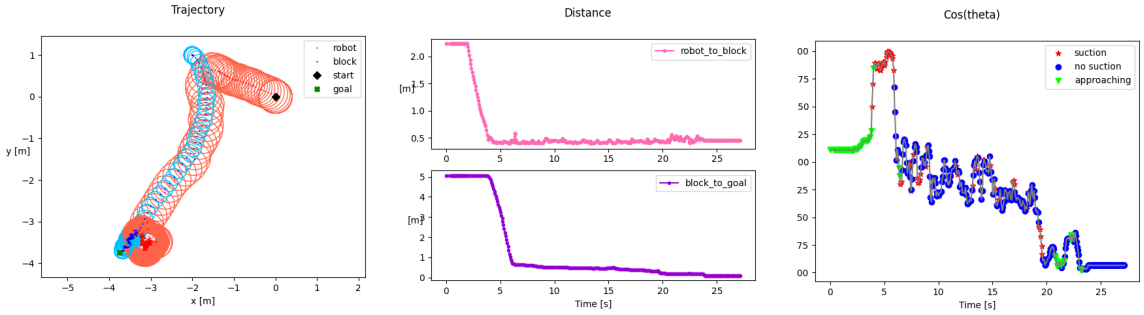


Figure A.5: Metrics of hybrid action.