Reactive Task and Motion Planning for Mobile Manipulation

Literature Report

Yuezhe Zhang

TU Delft

Reactive Task and Motion Planning for Mobile Manipulation

by

Yuezhe Zhang

Student number:	5390664	
Thesis committee:	Dr. J. Alonso-Mora,	TU Delft, main supervisor
	Ir. E. Trevisan,	TU Delft, daily supervisor
	Ir. C. Pezzato,	TU Delft, daily supervisor

Cover:	Image by Nicholas StHelier, found on Professor Daniela Rus's website
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Over the last decades, the task planning community has concentrated on solving planning problems in discrete domains from a high-level perspective and has developed many effective, domain-independent search algorithms to generate a plan skeleton or a sequence of actions for the robot to achieve some goals. Research in motion planning, on the other hand, has focused on generating collision-free trajectories in configuration space from a low-level perspective. However, considering a pre-computed action skeleton and collision-free trajectories separately is ineffective, especially when the robot is operating in unstructured human-centered environments and the high-level actions could vary depending on the low-level motion states and changing environments. Consequently, in order for the robot to complete complex tasks in unstructured environments, planning needs to formulate the discrete high-level actions and continuous low-level motions in an integrated way. This introduces the problem of *Task and Motion Planning* (TAMP). The state-of-the-art TAMP approaches consider planning in a static space that encompasses many objects. Some focus on designing interdependence layers that connect task planning and motion planning. However, less attention is paid to generating reactive behaviors in a dynamic and uncertain environment where unexpected disturbances may influence the planning solutions. Therefore, this literature survey reviews the state-of-the-art task and motion planning techniques for mobile manipulation, and special focus is paid to generating versatile reactive behaviors and efficient motion planning techniques in multi-task, contact-rich, uncertain, and dynamic environments. In the research proposal, the proposed control scheme could provide adaptability at the task level through online decision making, as well as low-level reactiveness by means of a parallel sampling motion planner.

Nomenclature

List of Abbreviations

AIF	Active Inference
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
DMPs	Dynamic Movement Primitives
EFE	Expected Free Energy
FEP	Free-Energy Principle
GPU	Graphics Processing Unit
H-CSP	Hybrid Constraint Satisfaction Problem
IL	Imitation Learning
LfD	Learning from Demonstration
LHS	Left Hand Side
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral Control
PDDL	Planning Domain Definition Language
POMDP	Partially Observable Markov Decision Process
PRM	Probabilistic Roadmap
RHS	Right Hand Side
RL	Reinforcement Learning
RRT	Rapidly-Exploring Random Tree
STN	Simple Temporal Network
STOMP	Stochastic Trajectory Optimization for Motion Planning
STRIPS	Standard Research Institute of Problem Solver
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 $P(o_\tau \mid s_\tau, \boldsymbol{A}) = Cat(\boldsymbol{A} s_\tau)$
$\boldsymbol{B}_{a_\tau} \in \mathbb{R}^{m \times m}$	Transition matrix, indicating the probability of state transition under certain action a_τ. The columns are categorical distribution $P(s_{\tau+1} \mid s_\tau, a_\tau) = Cat(\boldsymbol{B}_{a_\tau} s_\tau)$
$\boldsymbol{C} \in \mathbb{R}^r$	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

1 Introduction

This introductory chapter focuses on the motivation behind this literature review, 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 proceeds with the main research question which will guide us to investigate the topic throughout the entire survey. Finally, the outline of the document is given to show a clear structure.

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.

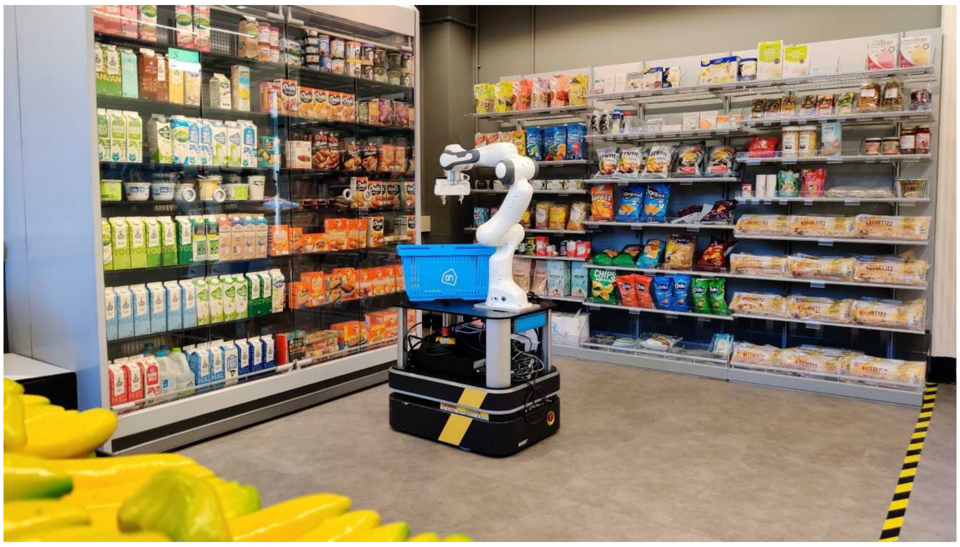

Figure 1.1: A mobile manipulator from the AI for Retail (AIR) Lab Delft [1]

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 literature survey, the focus is on the latter two aspects, namely task planning and motion planning, and the interdependence in between.

To narrow down the research scope, the problem we focus on is using a robot r (a mobile robot or a mobile manipulator) to perform different actions (pushing, pulling, picking, or placing) on multiple blocks $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ to the desired goal regions $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$. In addition, the robot should maintain some internal properties (battery level) so that it should go recharge at a recharged location l_r while performing current tasks. Unlike classical task and motion planning settings, the problem is considered to be in a *dynamically changing environment* where external unexpected changes can interfere. This includes the presence of dynamic obstacles $\mathcal{O} = \{o_1, \ldots, o_m\}$ and external actors that can freely move blocks and occlude the path of the robot during execution.

1.2. Background

As the literature survey focuses on task planning and motion planning, this section provides a brief overview of the fields that entail the topic of the survey. In each following chapter, related works and methods specific to each field are further discussed.

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. [2] provides a comprehensive guide to task planning from the AI perspective. Karpas and Magazzeni [3] 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 [4, 5] 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 [4–6]:

- 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 (AIF), which tries to explain how information is processed by the human brain [7]. The mathematical framework of AIF was constructed on the Free-Energy Principle (FEP) proposed by Karl Friston [8]. It has shown its potential in the domain of adaptive control such as controlling manipulators [9] and fault tolerant systems [10, 11] and also in the domain of symbolic reasoning [12]. 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.

1.2.2. Motion Planning

The problem of motion planning was formulated by Lozano-Pérez [13] as a search for paths in the robot's configuration space from one state to another without colliding with any obstacles. LaValle [14] 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 [15–17] or constrained optimization [18–20]. Besides, machine learning techniques [21, 22] have also been applied to generate a motor skill trajectory. A review of these three prominent methods is given in chapter 3.

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 [23–26]. 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.

Despite its appealing and elegant characteristics in theory, there are still some robustness issues when implementing and deploying MPPI in practice [27]. For example, as can be shown in Figure 1.2, due to the noisy and uncertain environments, the MPPI algorithm fails to capture the correct predicted dynamics and diverges so that the newly sampled control sequences all lie within the infeasible regions, which could lead to constraints violation. The reason behind the scene why MPPI fails when encountering unexpected disturbances, especially when using sparse objective information cost function is that the sampling-based method MPPI is still an iterative local search method, which means MPPI may get stuck in local minima, which highly relies on the initialization of the control sequences.

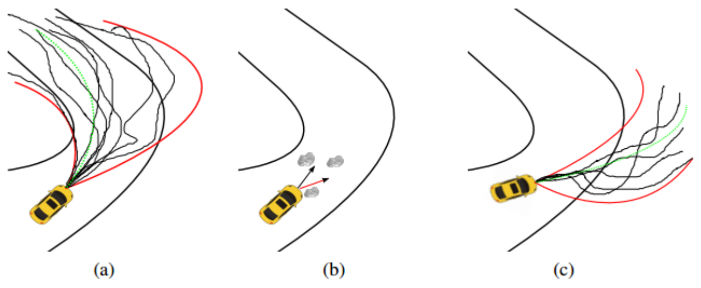

Figure 1.2: MPPI robustness issue [27]

To address this issue, prior works mainly applied an ancillary controller to track the output of the MPPI controller to keep the real trajectory as close as possible to the predicted trajectory. Authors in [27] used an iterative Linear Quadratic Gaussian (iLQG) and a Model Predictive Control based on Differential Dynamic Programming (MPC-DDP) separately as the second controller to track the nominal system state which was produced by MPPI. Authors in [28] utilized a tracking controller with L_1 augmentation to compensate for the distinction between the nominal trajectory and the true trajectory. However, these approaches separate the planning and control so that MPPI simply acts as a path planner. A recent study [29] has shown that MPPI can be incorporated with covariance steering theory by changing the terminal covariance constraint to generate adjustable trajectory sampling distributions, which could augment the performance when dealing with the unexpected disturbances and uncertainties.

However, less work focuses on massive parallel sampling in contact-rich environments, where the robot needs to reason about the effects of executing different control sequences through dynamic interaction with the environments.

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 key issue of TAMP problems is the interdependence of high-level task planning and low-level motion planning. For example, if a robot is programmed to pick up a product on a shelf and place it on a table, but the table has already been occupied by other objects, it is difficult to find satisfying solutions to place the product. In this situation, the robot needs to rearrange a new sequence of actions that will move the occupying objects before placing the product. This example demonstrates the importance of low-level geometry information in the selection of high-level actions.

Caelan Garrett et al. [30] characterized three predominant strategies integrating the discrete and continuous components of the TAMP problems: *sequence before satisfy*, where the action sequence is iteratively selected before trying to solve the constrained motion planning problem; *satisfy before sequence*, where trajectories are iteratively found for individual specific constraints before assembling these actions into complete plans; and *interleaved*, where planned action and satisfying constraints are added incrementally. The flowchart of the first two strategies is shown in Figure 1.3, and a detailed explanation of these methods will be elaborated in chapter 4.

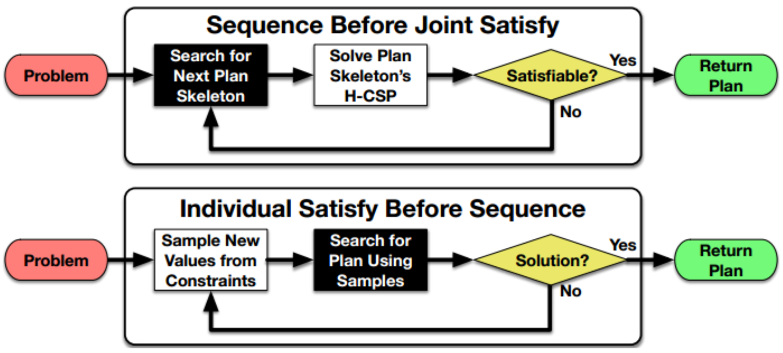
Sequence Before Joint Satisfy
Problem
Search for Next Plan Skeleton
Solve Plan Skeleton's H-CSP
Satisfiable?
Yes
No
Return Plan
Individual Satisfy Before Sequence
Problem
Sample New Values from Constraints
Search for Plan Using Samples
Solution?
Yes
No
Return Plan

Figure 1.3: Two strategies of TAMP framework [30]

However, the state-of-the-art TAMP approaches consider planning in an ample space that encompasses many static objects. They also focus on designing interdependence layers that connect task planning and motion planning. Less focus is paid to generating reactive behaviors in a dynamic and uncertain environment where unexpected disturbances may influence the planning solutions . In addition, there are still gaps in achieving high-level and low-level reactive control simultaneously.

1.3. Research Questions

The aforementioned research motivations and existing research gaps bring to the formulated question that this work aims to address:

"How can Task and Motion Planning (TAMP) incorporate versatile reactive behaviors and efficient motion planning techniques in multi-task, contact-rich, uncertain, and dynamic environments? "

The main question above can be divided into several sub-questions, whose answers will contribute to answering the main research question:

- *How does the mobile manipulator generate high-level reactive plans to cope with uncertainties and unexpected events?*
- *Once a high-level sequence of action has been decided, how can a mobile manipulator achieve efficient, versatile, and contact-rich manipulation motions in dynamic environments?*
- *How can the task planning and motion planning be incorporated to facilitate the interdependence?*

1.4. Outline

To help answer these questions, the survey is organized as follows.

Chapter 2 introduces the general approaches to task planning and mainly focuses on the approaches that fulfill the requirements of hierarchical deliberation and continual online planning. Comparisons between different approaches and formalisms are also discussed.

Chapter 3 provides a review of the state-of-the-art motion planning approaches underlining the benefits and limits of each approach.

Chapter 4 establishes the connection between task planning and motion planning, introducing some frameworks and special attention will be paid to the approaches that can handle disturbances and unexpected events.

Chapter 5 presents the research proposal including the proposed method and time schedule.

The final chapter summarizes the contents this survey presents and the conclusions are made about the original questions this survey aims to answer.

2
Task Planning

In subsection 1.2.1, two important properties that an intelligent actor should possess have been introduced. This chapter aims at surveying some fundamental task planning formalisms underlining their advantages and limitations. Moreover, special focus will be paid to the state-of-the-art techniques that will meet the requirements of the two properties and our application. Finally, discussions will be made about the comparison between state-of-the-art formalisms. Justification will also be provided for the choice in our case.

2.1. Why Task Planning

This chapter mainly reviews the topics and formalisms of robot task planning in order to achieve autonomous reactive behaviors. However, it is worth discussing at the very beginning why automated planning or task planning is important when robots are required to perform various tasks in our case. A brief overview of several high-level approaches to robot autonomous behavior was outlined by Geffner in [31], including programming-based, learning-based, and planning-based approaches.

In the programming-based approach, a human programmer is required to specify the actions that the robot needs to do. This is suitable in the case when the environments are often unchanged and the robots are required to perform repetitive behaviors, such as in the industrial line scenario. However, this is difficult when the environments are dynamically evolving and it thus requires the programmer to anticipate all the possible scenarios and preprogram all the tasks.

In the learning-based approach, the robot's behavior is induced via a learning algorithm from its own experiences (as in reinforcement learning) or from an expert (as in imitation learning). The learning-based approach has the greatest potential and flexibility. It does not require too much expert effort, but it only requires the user to provide a few more demonstrations as in imitation learning. However, it is limited in scope due to the lack of guarantees on the optimal policy or safety when exploring reinforcement learning and the lack of high-quality and quantity of data.

Despite the flexible applications the learning-based approach may achieve, it does not dispense with the need for effective planning-based methods. In fact, as pointed out in [4, 32], the planning-based robot can make adaptive plans to achieve very complex and changing tasks. It often assumes to have the model of the robot and the world, and the robot can use the model to reason the current situation and infer the next possible action, which will affect the world state. This approach is also highly interpretable compared with black box methods and has shown promising generalization abilities across different settings. The plans can enable the robots to flexibly explore the environments, adjust their action sequence and revise and optimize the intended plans. As such, reactive planning is considered with large significance in this chapter.

2.2. Related works

This section reviews the fundamental formalisms for robot task planning. A comprehensive overview in this field can be found in [2, 3]

2.2.1. Classical Formalism

Classical task planning is the simplest formalization of task planning which assumes that: *(a)* the world and the model is deterministic; *(b)* the state is fully observable; *(c)* the robot is the only agent that can influence and change the states of the world; *(d)* only instantaneous actions are involved. The formalism of classical task planning includes:

- A state space S, which contains a set of states of the world. States can be described by a set of variables, for example, if a robot is at the Table and holds nothing, and a product Milk is at the Shelf, this state can be defined as `{atRob = Table, holding = None, atMilk = Shelf}`. Each variable has assignments of values, which constitute a domain for this variable, for example, the variable `atMilk` can have possible locations `{Table, Shelf, Ground}`. Thus a state space can be described as the Cartesian product of each variable's domain $S = S_1 \times S_2 \times ... \times S_n$, where n is the number of variables.
- A set of transitions (or actions) between these states, $T \subseteq S \times S$, where each transition $t = \langle s, \hat{s} \rangle$ changes the system state from s to $\hat{s}$.
- An initial state $s_0 \in S$ and a set of goal states $G \subseteq S$.
- Preconditions `pre` describe the conditional state that an action can be performed.
- Effects `eff` describe the legal changes one action can bring to the world.
- The solution to a task planning problem is to find a sequence of transitions $t_1, t_2, \ldots, t_k$, that originates from the initial state s_0 and proceeds to the goal state $s_k \in G$, where the precondition of transition t_i is state s_{i-1} and its effect is state s_i.

This problem can be transformed into a graph-search problem, where the nodes represent states and directed edges represent transitions, and it can be solved using graph-search algorithms. However, the state space considered can be very large, due to the fact that each variable can have a large number of associated values, and thus the overall dimension of the state space. So it is essential to use an appropriate representation of a set of transitions.

STRIPS

The first classical task planning formalism that has been applied to a real robot is the so-called STRIPS (Standard Research Institute of Problem Solver) [33]. The robot Shakey [34] is shown in Figure 2.1. In this formalism, each action was modeled as a STRIPS operator with preconditions and effects, allowing the planner to update the state of the world after the action was performed. The generated high-level plan was later realized by the low-level actions. However, the issue lay in that Shakey assumed the problems satisfied the *downward refinement* property [35], which means every solution to the high-level plan could be transformed into a solution to the low-level action. This would lead to problems since no alternative high-level plan would be found automatically if low-level motions failed. Potential solutions to address this issue will be elaborated in chapter 4.

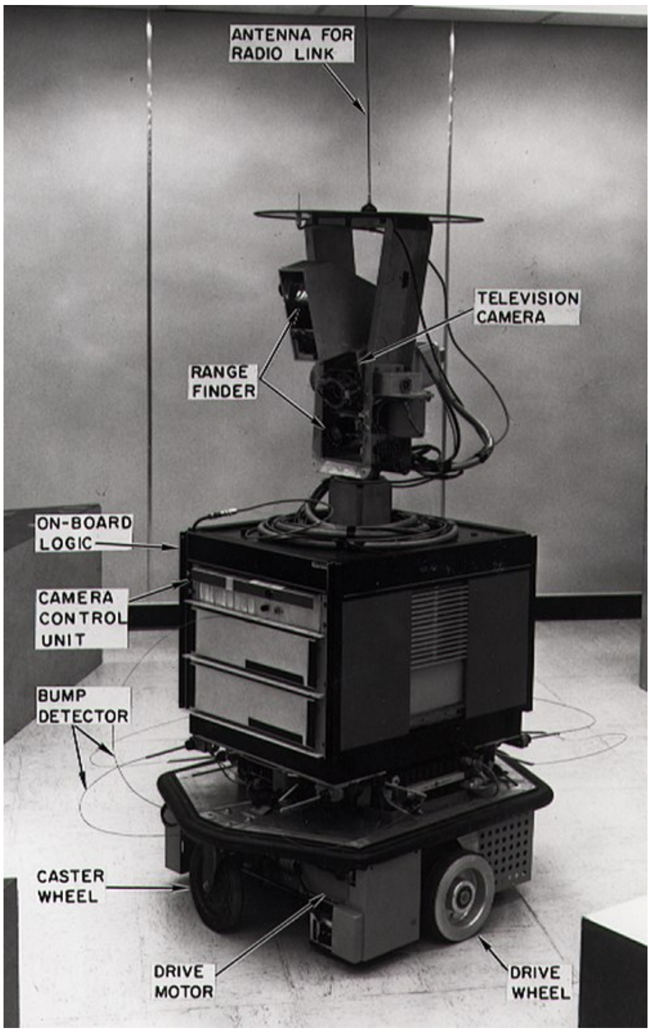

Figure 2.1: Shakey the robot [34]

PDDL

Inspired by STRIPS, the most widely-used and standard formalism *Planning Domain Definition Language* (PDDL) was created by Drew McDermott [36] to develop the domain-independent algorithms that can be used on any problems. PDDL separates the definition of a planning problem into two parts: the domain description and the problem description. The domain description can capture the common elements of all problems, while the problem description contains the specific elements of every problem. Thus several problems can be connected to the same domain descriptions. Several versions of PDDL have been developed, among which PDDL1.2 is the most basic syntax supported by various planners.

The contents of PDDL1.2 domain file contain:

- Definition of possible object types. For example, the object types in the retail store could be a robot, table, shelf, milk, hagelslag;
- Definition of predicates, which describe the logical relationships between objects. For example, the predicate `at(?m, ?t)` could indicate whether milk product `?m` is on the table `?t`;
- Definition of possible actions, which are a set of operator-schemas with parameters. In addition to the parameters, the actions also have preconditions and effects.

The contents of PDDL1.2 problem file contain:

- Definition of all possible objects in the world;

- Definition of the initial state, which is a list of true grounded propositions;
- Definition of goal state, which is a logical expression of facts that should be true or false in the desired state.

2.2.2. Temporal and Numeric Planning

PDDL1.2 was extended to PDDL2.1 by Fox and Long [37] to model temporal and numeric structures. The introduced numeric variables enable the modeling of time, energy, distance, and battery charge. The introduced plan metrics enable the plan to be evaluated quantitatively instead of satisfying the goals only. This is especially useful when PDDL2.1 is combined with a cost-optimal planner, which can help select action sequences with the least time duration or energy. The introduced durative actions allow the action to be associated with a starting time and a duration, which makes it flexible when some activities must be performed within time windows or concurrent activities should be considered.

The temporal plans can be expressed as a Simple Temporal Network (STN) [38], where the temporal constraints in the network can encode the minimum and maximum time that can pass between the events. Executors can thus use STN structure to arrange flexible time execution. The STN structure can be used directly by the planners such as the Advanced Planning and Scheduling Initiative (APSI) planner [39], the Extensible Universal Remote Operations Planning Architecture (EUROPA) [40], the IxTeT [41].

After generating the temporal plans, the robot is supposed to know what to do and when to do it, however, when executing these behaviors in practical scenarios, it raises issues regarding how to observe the deviation of the process and how to understand when the plan is invalid. The interaction between high-level planning and execution has been the central issue this survey wants to explore, and more literature will be surveyed and elaborated on from the perspective of integrated task planning and motion planning in chapter 4. While with respect to the temporal plan execution at this part, the approach [42] that computes the ”robustness envelope”, which is a concept of alternative action durations or resource consumption rates of given plans, to determine whether they are still valid despite delays should be noticed.

2.2.3. Cost-optimal Planning

The aforementioned formalisms focus on finding a plan that satisfies certain constraints and goals, however, they can not find a plan that is optimal. This introduces the problem of *cost-optimal planning*. Even though all these problems have been shown to be PSPACE-complete [43], in general, proving a plan is optimal is much harder than finding a plan.

Most cost-optimal planners leverage heuristic search [44] to traverse the state space graph in order to find a plan between the initial state and the goal state. In order to prove optimality, best-first search algorithms are used to explore all the states that cannot contribute to an optimal solution. Best-first search algorithms maintain two lists of search nodes, one with all the unexplored states, while the other with all the explored states whose successors have already been generated.

Dijkstra’s algorithm is a type of best-first search that explores the nodes in ascending order of value $g(s_n)$ [45], where the $g(s_n)$ of a search node s_n represent the cumulative costs from initial s_0 to s_n. However, this algorithm is computationally costly since it may explore an exponential number of nodes. In order to alleviate this issue, a *heuristic* $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$ can be combined to estimate the remaining cost of the current node. The remaining cost of a state s, $h^*(s)$ is the cost of a cost-optimal path from s to the goal state. A^* [46] explores the nodes with lowest $f(s_n) = g(s_n) + h(s_n)$ first so that no node with $f(n) > h^*(s_0)$ will be explored. If a heuristic is *consistent*, A^* will never re-explore a node, which guarantees optimality on expanding nodes. Besides, storing explored nodes in a list allows A^* easily prune duplicate nodes. These properties have made A^* the predominant algorithm for cost-optimal planning.

2.2.4. Top-k Planning

As was introduced in previous subsection, the objective of cost-optimal planning is to find a sequence of actions that leads to a goal with minimal cumulative action costs. However, this type of cost-optimal planning cannot provide other alternative plans, which could be beneficial in considering factors that are difficult to model or may have changed when the plan is executed, such as user preferences and environmental influences. Thus top-k planning is referred to as the problem of determining not only the single best plan but also a set of k best plans [47, 48]. A top-k planner can generate and test high-quality plans, which can be applied in areas such as goal recognition [49], explanation generation [50], and even machine learning algorithms [51, 52].

Definition Top-k planning can be described as the problem of determining a set of plans $P \in P_\Pi$ given a planning task Π and an integer k such that:

1. There exists no plan $\pi' \in P_\Pi$ with $\pi' \notin P$ that is cheaper than some plan $\pi \in P$;
2. $|P| = k$ if $|P_\Pi| \le k$, and $|P| = |P_\Pi|$, otherwise.

where $|P|$ means the size of the set of plans. An algorithm for top-k planning is called *sound*, if and only if it satisfies condition 1. This definition of soundness also implies optimality, since a top-k planner should only give *top* plans with optimal cost. An algorithm for top-k planning is called *complete*, if and only if when it satisfies condition 2.

Symbolic Search

Compared with explicit search, which is the most common way of searching in planning, symbolic search makes use of concise data structures, such as Binary Decision Diagrams (BDDs) [53], to represent a set of states. For example, a set of states $S \subseteq \mathbb{S}$ can be represented by its corresponding Boolean function $\Omega_S : \mathbb{S} \to \{True, False\}$, such that $\Omega_S(s) = True$ if $s \in S$ and $\Omega_S(s) = False$ otherwise. In addition, operators can be represented as transition relations over states, which enable efficient manipulation over states. For example, a set of operators $O \subseteq \mathbb{O}$ can be represented as a transition containing the set of state pairs $(s, s^{'})$ such that $s^{'}$ is reachable after performing an operator $o \in O$. Therefore, the symbolic search can expand and generate whole sets of states at once instead of single states.

Symbolic Dijkstra search [45] in the forward direction (progression) starts with the transition function of the initial state and iteratively computes the successors until the goal states are reached. Symbolic backward search (regression) can be performed from goal states and applies the preimage operation until the initial state is found. It is also possible to combine these two strategies to form a bidirectional search in symbolic planners [54].

Symbolic Top-k Planning

Symbolic planning was first used for top-k planning in [55], called SYM-K. It has been proven sound and complete for top-k planning. The reason why symbolic search can be applied to top-k planning is that:

- As already mentioned, the symbolic search can expand and generate whole sets of states at once instead of single states;
- Thus, multiple plans can be found when exploring states instead of finding only the optimal plan.

Although SYM-K performs a variant of symbolic Dijkstra search (SYM-DIJ), which is similar to [56, 57], the differences from normal SYM-DIJ lie in that:

- Once a goal state is explored, all plans leading to that goal state are reconstructed;
- States are not closed, which means all generated states are stored in an open list;
- SYM-K terminates if either k plans are found or the open list contains only states that have already been expanded at least once and are not part of a goal path induced by a plan already found.

The proposed SYM-K algorithm was implemented in the $SYMBA^*$ planner [56], which is built based on the FAST DOWNARD planning system [58]. The performance in k-convergence was evaluated and compared with modern top-k planning algorithms, such as K^* and $FORBID-K$ [48]. It showed that SYM-K performed better than other approaches both in small k and large k.

2.2.5. Planning with Uncertainty

The aforementioned formalisms only consider deterministic problems with fully observed information, however, these are not applicable when the real robot actions could fail and only partially observed information could be obtained. Planning problems related to nondeterministic dynamics and partial observability can be formalized as a Partially Observable Markov Decision Process (POMDP) by Karl Johan Åström [59]. It was later adopted to artificial intelligence and automated planning by Leslie P. Kaelbling, and Michael L Littman [60]. Recently, Hanna Kurniawati [61] provided a review about POMDP, emphasizing the idea of sampling-based solvers to alleviate the computational issues related to the application of robots.

POMDP

Definition The POMDP formulation can be defined as a 6-tuple $\langle S, A, O, T, Z, R \rangle$, where the meanings of the variables can be shown as:

- S describes the state space, containing the states of the robot and the environment;
- A describes the action space, containing the actions a robot can perform;
- O describes the observation space, containing the observations a robot can obtain;
- $T(s, a, s')$ describes the transition function. It can be expressed as a conditional probability function $P(s'|s, a)$, meaning the probability of a robot being at state s' after performing action a at state s;
- $Z(s', a, o)$ describes the observation function. It can be expressed as a conditional probability function $P(o|s', a)$, meaning the probability of a robot perceiving observation o after performing action a at state s;
- R describes the immediate reward function.

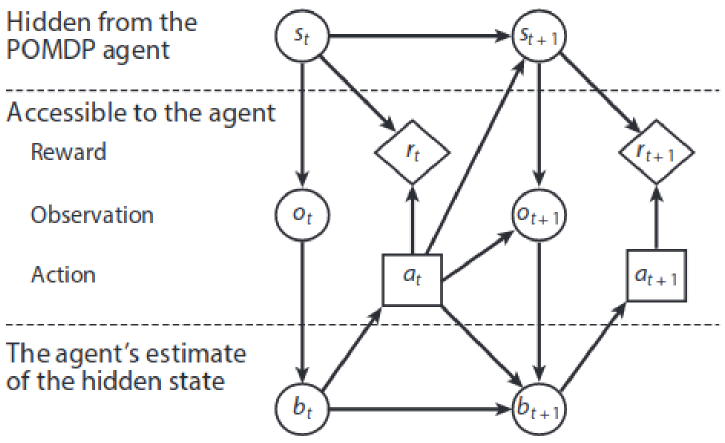

Figure 2.2: Illustration of a POMDP process [61]

An illustration of a POMDP process is shown in Figure 2.2. At timestep t, the robot is in a partial-observable state s_t and holds a belief of its state b_t. The robot will infer the best action a from current belief b_t based on the value function. The reward will also be computed via the reward function based on the state action pair. The hidden state s_t will move to state s_{t+1} after performing action a_t based on the transition function. The observation o_{t+1} towards s_{t+1} can be updated after action a_t based on the observation function. Then a new belief b_{t+1} will be updated based on the previous belief b_t, the performed action a_t and the observation o_{t+1} via Bayesian inference.

Solving a POMDP problem requires an objective function to find an optimal policy from current belief. An infinite horizon objective function can be formulated as Equation 2.1, based on the expected total discounted reward and the expected immediate reward:

$$V^*(b) = \max_{a \in A} \left(R(b,a) + \gamma \sum_{o \in O} P(o|b,a) V^*(\tau(b,a,o)) \right) \tag{2.1}$$

where $R(b,a)$ is the expected immediate reward based on the current belief b and potential action a, and $\tau(b,a,o)$ updates the belief b after performing action a and perceiving o, and the discounted factor γ takes into account the fact the reward for an agent in infinite horizon may be minor. Thus finding the best action from the current belief b involves the estimate of the expected future reward $J(b,a) = \sum_{o \in O} P(o|b,a)V^*(\tau(b,a,o))$ and optimization of the Q-value $Q(b,a) = R(b,a) + \gamma J(b,a)$ for belief b.

Definition A finite horizon value function can be defined as:

$$V_t^*(b) = \max_{a \in A} \left(R(b,a) + \gamma \sum_{o \in O} P(o|b,a) V_{t-1}^*(\tau(b,a,o)) \right) \tag{2.2}$$

where $V_0^*(b) = \max_{a \in A}(R(b,a))$, $V_t^*(b)$ is the reward of belief b when the robot can move for t future steps with $t < T$.

Finding an optimal solution to a POMDP problem is PSPACE-complete [62], making it computationally intractable to solve exactly for real robot systems. It is especially difficult when a large state space, long planning horizon, large observation space, large action space, and complex transition dynamics involve. A sampling-based approximate POMDP solver [63] was proposed to compute good policies on a large (870 states) robot domain. Sampling-based approximate POMDP solvers relax the optimality requirement and restrict the problem to start from a given initial belief. A general sampling-based approximate approach can be shown in Algorithm 1:

Algorithm 1 A general sampling-based approximate POMDP approach [61]

```
Step 1: start from given initial belief $b_0$ and initialize policy $\pi$
repeat
  Step 2: sample belief;
  Step 3: estimate the values of the sampled beliefs via a combination of heuris-
tics and backup operation;
  Step 4: update policy $\pi$;
until stopping criteria meet
```

At step 2, sampling belief from b_0 is to use action a and observation o to generate trees to form new belief b', which can be described as $b' = \tau(b_0, a, o)$. Thus the newly sampled belief b' is connected with b via the connection of action a and observation o. It will then iteratively expand the belief trees. At step 3, the backup operation means estimating the expected reward of the sampled belief b_0 relies on the expected value of its descendants b'. Some functions are proposed to iteratively trace from the leaves of the belief tree to the root of the tree in order to improve the estimate of the Q-values of performing different actions. Then the best action is selected to update the policy π. The whole process will come to an end when the stopping criteria satisfy.

Most sampling-based POMDP solvers can be divided into offline and online solvers. Offline solvers compute an offline optimal policy before execution. The robot only needs to estimate and update its belief b in real-time and execute the action $\pi(b)$. In contrast, online solvers can do policy searching and action execution in real-time.

Bayesian Brain Hypothesis

Another well-known theory for reasoning with uncertainty is Bayesian inference and Bayesian brain hypothesis [64, 65]. The first claim of the Bayesian brain hypothesis is that the brain is equipped with an *internal generative model* of the environment, which generates *sensory observations* o from *hidden states* s. In the internal model, hidden states s are drawn from a prior distribution $p(s)$ and the sensory observations are drawn from an observation distribution conditional on the hidden states $p(o|s)$. The mapping from the hidden states to the observations might be noisy and thus can be quantified by the *likelihood*, which describes the probability of certain observations given a set of hidden states.

The second claim of the Bayesian brain hypothesis is that the prior distribution and likelihood are combined to infer the hidden states given the observation, which can be described by Bayes' rule:

$$p(s|o) = \frac{p(o|s)p(s)}{p(o)} \tag{2.3}$$

where $p(s|o)$ is the posterior distribution and $p(o) = \sum_s p(o|s)p(s)$ is the *marginal likelihood*. It can be thought of as inverting the internal model to compute the belief about the hidden states given the observations.

The Bayesian brain hypothesis can be extended to situations where an agent can take actions and thus influence the observations. The action is taken from a policy π, which maps from observations to an action distribution. In this case, the agent will choose a policy that maximizes *information gain*:

$$I(\pi) = \sum_o p(o|\pi) KL[p(s|o,\pi)||p(s|\pi)] \tag{2.4}$$

where o denotes the future observation and KL denotes the Kullback-Leibler (KL) divergence, which measures the difference between the two distributions:

$$KL[p(s|o,\pi)||p(s|\pi)] = \sum_s p(s|o,\pi) \log \frac{p(s|o,\pi)}{p(s|\pi)} \tag{2.5}$$

The expression of information gain $I(\pi)$ is equivalent to Bayesian surprise [66], and information maximization has been studied extensively in the neuroscientific literature [67–69]. It can also be considered as a special case of Bayesian decision theory, where the information gain corresponds to an observation s and its utility $u(o) = KL[p(s|o,\pi)||p(s|\pi)]$. The agent seeks to maximize the expected utility:

$$\mathbb{E}[u(o)|\pi] = \sum_o p(o|\pi)u(o) \tag{2.6}$$

It can be generalized to sequential decision problems, whose return is represented by discounted cumulative utility:

$$R(\boldsymbol{o}) = u(o_1) + \gamma u(o_2) + \gamma^2 u(o_3) + \dots \tag{2.7}$$

where the bold notation $\boldsymbol{o} = [o_1, o_2, \dots]$ represents the time sequence of observations. And the agent's goal is to maximize the expected return:

$$\mathbb{E}[R(\boldsymbol{o})|\pi] = \sum_{\boldsymbol{o}} p(\boldsymbol{o}|\pi)R(\boldsymbol{o}) \tag{2.8}$$

The sequential decision problems involve the trade-off between exploration and exploitation, where exploration means the agent collecting information to reduce uncertainty, and exploitation means taking actions to yield more rewards. This means the preferences for information will arise in the setting and thus need not be built explicitly into the utility function.

It has been summarized in [70] that there are some issues related to the Bayesian brain hypothesis:

- There are many empirical deviations from its claims. [71] pointed out that humans often exhibit biases in judgment that are biased from Bayesian inference, and these biases can be attributed to heuristics or limitations in the cognitive process. [72] counteracted assertions about human actions in an optimal Bayesian way by bringing together the literature on suboptimal perceptual decision making and focusing on building a science of observations that attempts to account for all types of behaviors.

- The Bayesian brain hypothesis does not make any specific claims about the priors and likelihood. Instead, it concerns more about the consistency of beliefs: an agent will convert the prior beliefs into posterior beliefs according to Bayes's rule.

2.3. Free-Energy Principle

Among these theories, the Bayesian brain hypothesis has been extended by Karl Friston to the free-energy principle [8] 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. [70] clarifies the connections between FEP and earlier unifying ideas such as Bayesian inference, predictive coding, and active learning in a more comprehensive way.

2.3.1. 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:

$$-\ln p(o) \tag{2.9}$$

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.

2.3.2. 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 as shown in Equation 2.3 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:

$$q^*(s) = \underset{q(s)}{\text{argmin}}\, KL[q(s)||p(s|o)] \tag{2.10}$$

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:

$$\begin{aligned} KL[q(s)||p(s|o)] &= \sum_s q(s) \log \frac{q(s)}{p(s|o)} \\ &= \sum_s q(s) \log \frac{q(s)p(o)}{p(s,o)} \\ &= \sum_s q(s) \left[\log \frac{q(s)}{p(s,o)} + \log p(o)\right] \\ &= \sum_s q(s) \log \frac{q(s)}{p(s,o)} + \log p(o) \sum_s q(s) \\ &= F[q(s)] + \log p(o) \geq 0 \end{aligned} \tag{2.11}$$

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

$$F[q(s)] = \sum_s q(s) \log \frac{q(s)}{p(s,o)} \tag{2.12}$$

The free energy is equivalent to the negative of the *evidence lower bound*, which is common in the machine learning literature [73]. It can be derived from Equation 2.11 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 2.9:

$$F[q(s)] \geq -\log p(o) \tag{2.13}$$

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.

2.3.3. 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:

$$p(s|o) \approx q(s) = \prod_i q_i(s_i) \tag{2.14}$$

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:

$$q(s) \sim \mathcal{N}(s; \mu, \Sigma) \tag{2.15}$$

2.4. Active Inference

Inheriting from the free-energy principle, Active Inference (AIF) was proposed by Karl Friston for describing and explaining how adaptive systems (biological or artificial) conduct perception, learning, planning, and decision making [7, 74]. 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 [75].

2.4.1. 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:

$$P(\overline{o}, \overline{s}, \boldsymbol{\eta}, \pi) = P(\overline{o}|\overline{s}, \boldsymbol{\eta}, \pi)P(\overline{s}|\boldsymbol{\eta}, \pi)P(\boldsymbol{\eta}|\pi)P(\pi) \tag{2.16}$$

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:

$$P(\overline{o}|\overline{s}, \boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(o_\tau|s_\tau) \tag{2.17}$$

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

$$P(\overline{s}|\boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi) \tag{2.18}$$

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

$$P(\boldsymbol{\eta}|\pi) = P(\boldsymbol{A})P(\boldsymbol{B})P(\boldsymbol{D}) \tag{2.19}$$

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 2.16, Equation 2.17, Equation 2.18, Equation 2.19, the generative model can be derived as:

$$P(\overline{o}, \overline{s}, \boldsymbol{\eta}, \pi) = P(\overline{o}, \overline{s}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \pi) = P(\pi)P(\boldsymbol{A})P(\boldsymbol{B})P(\boldsymbol{D}) \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi)P(o_\tau|s_\tau) \tag{2.20}$$

2.4.2. 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:

$$F(\pi) = \sum_{\tau=1}^{T} s_\tau^{\pi\top} \left[\ln s_\tau^\pi - \ln \boldsymbol{B}_{a_{\tau-1}} s_{\tau-1}^\pi - \ln \boldsymbol{A}^\top \boldsymbol{o}_\tau\right] \tag{2.21}$$

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

2.4.3. 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:

$$\begin{aligned} s_{\tau=1}^\pi &= \sigma\left(\ln \boldsymbol{D} + \ln \boldsymbol{B}_{a_\tau}^\top s_{\tau+1}^\pi + \ln \boldsymbol{A}^\top o_\tau\right) \\ s_{1<\tau<T}^\pi &= \sigma\left(\ln \boldsymbol{B}_{a_{\tau-1}} s_{\tau-1}^\pi + \ln \boldsymbol{B}_{a_\tau}^\top s_{\tau+1}^\pi + \ln \boldsymbol{A}^\top o_\tau\right) \\ s_{\tau=T}^\pi &= \sigma\left(\ln \boldsymbol{B}_{a_{\tau-1}} s_{\tau-1}^\pi + \ln \boldsymbol{A}^\top o_\tau\right) \end{aligned} \tag{2.22}$$

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

2.4.4. 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:

$$G(\pi,\tau) = o_\tau^{\pi\top}\left[\ln o_\tau^\pi - \ln C\right] - diag\left(A^\top \ln A\right) s_\tau^\pi \tag{2.23}$$

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

2.4.5. 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:

$$\boldsymbol{\pi} = \sigma\left(-\boldsymbol{G}_\pi - \boldsymbol{F}_\pi\right) \tag{2.24}$$

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_τ^π, the overall probability distribution for the states can be computed through Bayesian Model Average:

$$s_\tau = \sum_i s_\tau^{\pi_i} \boldsymbol{\pi}_i, i \in \{1, \ldots, p\} \tag{2.25}$$

where $s_\tau^{\pi_i}$ 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:

$$\begin{aligned} \lambda &= \text{argmax}\left([\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_p]\right) \\ a_\tau &= \pi_\lambda(\tau = 1) \end{aligned} \tag{2.26}$$

where λ is the index of the most likely plan.

The active inference algorithm [75] can be summarised as pseudo-code in Algorithm 2. 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 2 Action selection with active inference [75]

```
1: Set $\boldsymbol{C}$                                          ▷ Prior preferences
2: for $\tau = 1$ to $T$ do
3:     If not specified, get state from $\boldsymbol{D}$ if $\tau == 1$
4:     If not specified, get observation from $\boldsymbol{A}$
5:     Compute $F$ for each plan                                 ▷ Equation 2.21
6:     Update posterior state $s_\tau^\pi$                       ▷ Equation 2.22
7:     Compute $G$ for each plan                                 ▷ Equation 2.23
8:     Bayesian model averaging                                  ▷ Equation 2.25
9:     Action selection                                          ▷ Equation 2.26
10: end for
11: Return $a$                                                   ▷ Preferred action
```

2.5. 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 [76–78]. 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 [79] 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 [80], 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 [7] 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 [81]. 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 [82].

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 [81].

2.6. Summary

This chapter first explains the necessity of model-based task-planning methods in our research. This choice is due to its promising feature of generalization abilities across different problem settings and its ability to reason the effect of every action. Then, it surveys some fundamental topics of robot task planning, including the classical STRIPS and PDDL, temporal planning, cost-optimal planning, top-k planning, and planning with uncertainty. Among these theories, the Bayesian brain hypothesis has been extended to the free-energy principle to explain how adaptive systems conduct cognitive learning and planning, which is an ideal technique for our application with regard to hierarchical deliberation and continual online reasoning. It has been further developed into a unifying mathematical framework called Active Inference. Finally, the analysis of the state-of-the-art AIF and other formalisms, such as POMDP and RL, highlights its main advantages, namely the efficiency in computing probabilistic plans, the flexibility in encoding parameterized habits and preferences, and the essence of intrinsic informational objectives.

3 Motion Planning

This chapter aims at covering the typical motion planning approaches for high-dimensional systems, underlining the advantages and drawbacks of the current algorithms. Special attention will be paid to the approaches that can achieve efficient multiple contact-rich manipulation skills in dynamic environments. Finally, discussions will be made about the comparison between the state-of-art methods and what gap lies to be solved in our application.

3.1. Related Works

This section surveys the fundamental formalism and the most popular methods for robot motion planning, which can be divided into sampling-based, optimization-based, and learning-based methods. A comprehensive introduction to this field can be found in [14].

3.1.1. Classical Formalism

The motion planning problem for a robot with d degrees of freedom can be described as finding a trajectory for a point indicating the robot's configuration through a d-dimensional configuration space. It can be more formally specified by a configuration space $\mathcal{Q} \subset \mathbb{R}^d$, a constraint $F : \mathcal{Q} \rightarrow \{0, 1\}$, an initial configuration $q_0 \in \mathcal{Q}$ and a set of goal configurations $Q_* \subset \mathcal{Q}$. The feasible configuration space is a subset of the configuration space $\mathcal{Q}$ that satisfies the constraint $Q_F = \{q \in \mathcal{Q} | F(q) = 1\}$. The objective is to find a continuous path $\tau : [0, 1] \rightarrow \mathcal{Q}$ such that $\tau(0) = q_0, \tau(1) \in Q_*$, and $\forall \lambda \in [0, 1], \tau(\lambda) \in Q_F$. The simplest motion planning problems require the robot to move through space without colliding, where the free-space motion is constrained by $F(q)$.

Motion planning is PSPACE-hard [83], which suggests that, in the worst case, exponential time is required for any algorithm to solve the problem. However, there are exact algorithms that leverage algebraic geometry to solve problems using only polynomial space to prove it is PSPACE-complete [84]. In addition, there are three most widely used approaches to solve the motion planning problem: sampling-based methods [15–17], optimization-based methods [18–20], and learning-based methods [21, 22].

3.1.2. Sampling-based Methods

Sampling-based motion planning methods are developed to avoid the explicit construction of the obstacle space, instead, the configuration space is discretized by sampling, which decreases the complexity of the motion planning problem [85]. A graph is created by discretizing the configuration space via sampling vertices, collision checking, and edge connection. Each sampled vertex represents a possible robot configuration, which will be rejected if it hits or gets close to obstacles via the information provided by the collision-checking module, then the collision-free edge will be connected between vertices from the obstacle-free space. Two subcategories can be distinguished among the sampling-based methods: single-query and multi-query algorithms. A single-query algorithm, such as Rapidly-Exploring Random Tree (RRT) [16, 17], creates a graph specifically for the start and goal configuration, if a new start and goal configuration is updated, a new graph needs to be created. A multi-query algorithm, such as Probabilistic Roadmap (PRM) [15], constructs a graph that can be used for multiple queries. Even though multiple-query methods achieve good performance in highly structured environments, such as factory floors, most online planning problems do not require multiple queries, since, for instance, the robot moves from one environment to another, or the environment is not known as a priori. Moreover, in some applications, computing a roadmap as a priori may be computationally challenging or even infeasible.

The speed and effectiveness of sampling-based motion planning methods can be improved substantially thanks to the improvements in algorithms and hardware computing power [86–89]

The main advantages of sampling-based motion planning methods are:

- they are computationally efficient, especially in high-dimensional spaces or multi-obstacle scenarios, because they only rely on the collision checking module without the need of constructing an explicit representation of the environment. The collision checking can provide information about the feasibility of candidate trajectories;
- they are mostly probabilistic complete, which means if there exists a solution, the algorithm will find the solution as the number of the samples increases to infinity;
- some (PRM*, RRT*) are proven to be asymptotically optimal. Asymptotic optimality is important for many applications that require high-quality solutions and gives the user confidence that increasing the computational hardware applied to the problem will result in better and better solutions that improve toward optimality.

The main disadvantages of sampling-based motion planning methods are:

- the generated paths are often jerky, which requires optimization skills to smooth and shorten the computed trajectories;
- considerable computational effort is spent in creating and connecting samples in the configuration space that might not be relevant to the task.

3.1.3. Optimization-based Methods

Optimization-based methods formulate the motion planning problem to minimize the length of a trajectory encoded as a sequence of states and controls subject to constraints and obstacles. Optimization plays two important roles in motion planning: (a) it can be used at the post-processing step to smooth and shorten trajectories generated by other planning methods such as sampling-based methods. (b) it can be used to compute locally optimal, collision-free trajectories from scratch.

An example of an optimization-based method for high-dimensional robot application is Covariant Hamiltonian Optimization for Motion Planning (CHOMP), which uses gradients of smoothness and obstacle costs to improve the quality of an initial trajectory iteratively [18]. CHOMP demonstrates computational efficiency and convergence in environments containing obstacles. The drawbacks of CHOMP are that it is only applicable if the gradients of the cost functions are available and that gradient-based methods are prone to local minima. Stochastic Trajectory Optimization for Motion Planning (STOMP) uses a cost function similar to the one used in CHOMP, besides, it can deal with cost functions for which gradients are not available and overcomes local minima due to its stochastic nature [19]. STOMP creates noisy trajectories around an initially created trajectory to find a trajectory with lower costs. Both CHOMP and STOMP take the smoothness of the path and obstacle avoidance into account. The success rate and the number of iterations to success are improved by STOMP, while the planning time for CHOMP is lower. The drawbacks of CHOMP and STOMP are that they need a large number of trajectory states to reason about small obstacles and many constraints.

TrajOpt is another trajectory optimization approach for solving robot motion planning problems [20], which is formulated as a sequential convex optimization with l_1 penalty terms for satisfying constraints, including no-collision constraint in terms of the signed distance and continuous-time no-collision constraint in the form of support mapping representation. A repeat convex approximation to the problem is constructed and then optimized. In particular, the objective and inequality constraint functions are approximated by convex functions that are compatible with a Quadratic Program (QP) solver, and the nonlinear equality constraint functions are approximated by affine functions. The ability to add new constraints and costs to the optimization problem allows the approach to tackle a larger range of motion planning problems, including planning for underactuated, non-holonomic systems. It is shown that the algorithm can efficiently compute locally optimal, collision-free trajectories from scratch using infeasible trajectory initializations as opposed to smoothing a previously computed collision-free trajectory.

The main advantages of optimization-based motion planning methods are:

- the algorithm that plans in continuous space is optimal and complete if the algorithm is not prone to local minima;
- the algorithm is computationally efficient due to the advances of optimization solvers.

The main drawback of optimization-based motion planning methods is:

- obstacles, state constraints, and control constraints often require solving a non-convex, discontinuous constrained optimization problem, which is hard to solve directly and exactly;

It should be mentioned that the optimization-based method is not a replacement for sampling-based motion planning methods. It is not expected to find solutions to difficult planning problems, such as bug traps or maze pathfinding, and is not guaranteed to find a solution if one exists. Sampling-based motion planning methods such as RRTs or PRMs could be used to compute a feasible initialization that could be used to seed the optimization approach.

3.1.4. Learning-based Methods

Learning from Demonstration (LfD) has been an active research area that leverages machine learning techniques to obtain trajectory representations of a motor skill. Some widespread approaches include Dynamic Movement Primitives (DMPs) [21], primitives based on Gaussian mixture models [22], and Probabilistic Movement Primitives (ProMPs) [90]. These approaches have shown some generalization capabilities and have been used for many manipulation tasks including T-ball batting [91], the ball-in-a-cup game [92], robot table tennis [93], and performing a golf swing [94]. LfD has also been used to enable the robot to manipulate deformable objects. As shown in [95], learning from the demonstration was combined with force control strategies to learn force-based behaviors from demonstrations. However, learning skills for deformable objects encounter issues at a semantic level as discussed in [96], such as understanding the meaning of folded or wrapped to determine whether a configuration is a valid goal.

Reinforcement learning has also been used to learn and improve primitives from demonstrations [97]. Policy search methods have shown potential for learning in real robots [98], which gain higher rewards after initially demonstrated policy. However, it requires a large number of trials, which is often impractical or expensive for real robot systems. In addition, LfD relies on the quality of the demonstrations and it would be difficult if humans could not provide useful demonstrations in hazardous environments.

The main advantages of learning-based methods are:

- Learning from demonstrations would make the learning process faster;
- It would be user-friendly. The methods would enhance the application of robots in human daily environments.

The main limitations of learning-based methods are:

- It requires a large amount of these demonstrations or trials with high quality, which will influence the performance of learned skills;
- Some of the learned skills might be sub-optimal or ambiguous.

3.2. Information-theoretic Framework

Despite all the progress in the aforementioned methods, generalization always remains a major challenge, especially when the robots work in dynamic and uncertain environments and thus must react to new situations with intelligent decision making and fast execution. Model Predictive Control (MPC) [99–102] 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 [23–26] to optimize for non-convex, non-linear dynamics, and high-dimensional systems. MPPI [23] was originally based on a stochastic optimal control framework and was solved by path integral control theory [103]. However, this approach is only applicable to control-affine systems. In order to scale to a large class of stochastic systems, information-theoretic MPPI [25, 26] 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 review, we focus on the latter approach.

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

3.2.1. General Control Scheme

Consider a general discrete-time stochastic dynamical system:

$$\begin{aligned} x_{t+1} &= F(x_t, v_t) \\ v_t &\sim \mathcal{N}(u_t, \Sigma) \end{aligned} \tag{3.1}$$

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:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m|\Sigma|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2} v_t^\top \Sigma^{-1} v_t\right) \tag{3.2}$$

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

$$q(V|U,\Sigma) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m|\Sigma|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(v_t - u_t)^\top \Sigma^{-1}(v_t - u_t)\right) \tag{3.3}$$

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:

$$C(x_1, x_2, \ldots, x_T) = \phi(x_T) + \sum_{t=1}^{T-1} q(x_t) \tag{3.4}$$

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

$$S(V, x_0) = C \circ G_{x_0} \tag{3.5}$$

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

3.2.2. Free energy and Relative Entropy Inequalities

In contrast to the description of free energy in chapter 2, 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:

$$F(S, \mathbb{P}, x_0, \lambda) = -\lambda \log \left(\mathbb{E}_{\mathbb{P}} \left[\exp \left(-\frac{1}{\lambda} S(V) \right) \right] \right) \tag{3.6}$$

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. It has already been shown in Equation 2.5 for discrete probability distributions. Here a more general definition is given.

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

$$KL(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}} \left[\log \left(\frac{q(V)}{p(V)} \right) \right] \tag{3.7}$$

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:

$$F(S, \mathbb{P}, x_0, \lambda) \leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} [S(V)] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) \tag{3.8}$$

Proof. Start from the free energy of a control systemEquation 3.6. After introducing a second measure $\mathbb{Q}_{U,\Sigma}$, it can be rewritten with respect to $\mathbb{Q}_{U,\Sigma}$:

$$\begin{aligned} F &= -\lambda \log \left(\mathbb{E}_{\mathbb{P}} \left[\exp \left(-\frac{1}{\lambda} S(V) \right) \right] \right) \\ &= -\lambda \log \left(\mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\exp \left(-\frac{1}{\lambda} S(V) \right) \frac{p(V)}{q(V|U,\Sigma)} \right] \right) \end{aligned} \tag{3.9}$$

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:

$$\begin{aligned}
F &= -\lambda \log\left(\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\right]\right) \\
&\leq -\lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\
&= -\lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[-\frac{1}{\lambda}S(V) + \log\left(\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] - \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{p(V)}{q(V|U,\Sigma)}\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{q(V|U,\Sigma)}{p(V)}\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P})
\end{aligned} \tag{3.10}$$

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.

3.2.3. 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:

$$U^* = \underset{U\in\mathcal{U}}{\text{argmin}}\, \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T) + \sum_{t=0}^{T-1}\mathcal{L}(x_t, u_t)\right] \tag{3.11}$$

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 3.4, and a control cost that is a quadratic function with affine term β:

$$\mathcal{L}(x_t, u_t) = q(x_t) + \frac{\lambda}{2}\left(u_t^\top \Sigma^{-1} u_t + \beta_t^\top u_t\right) \tag{3.12}$$

Now that the density functions of the base distribution $\mathbb{P}$ and controlled distribution $\mathbb{Q}_{U,\Sigma}$ are already given by Equation 3.2 and Equation 3.3, and the KL-Divergence term in Equation 3.8 can be simplified as:

$$\begin{aligned}
KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\frac{q(V|U,\Sigma)}{p(V)}\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\prod_{t=0}^{T-1}\exp\left(-\frac{1}{2}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t) + \frac{1}{2}v_t^\top\Sigma^{-1}v_t\right)\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\log\left(\prod_{t=0}^{T-1}\exp\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t + \frac{1}{2}u_t^\top\Sigma^{-1}v_t + \frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right)\right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t + \frac{1}{2}u_t^\top\Sigma^{-1}v_t + \frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right]
\end{aligned} \tag{3.13}$$

Thus the free energy and relative entropy inequality Equation 3.8 can be reduced to:

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\le \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda KL(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t + \frac{1}{2}u_t^\top\Sigma^{-1}v_t + \frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right] \end{aligned} \tag{3.14}$$

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

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\le \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^\top\Sigma^{-1}u_t + \frac{1}{2}u_t^\top\Sigma^{-1}v_t + \frac{1}{2}v_t^\top\Sigma^{-1}u_t\right)\right] \\ &\le \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V)\right] + \lambda \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\sum_{t=0}^{T-1}\left(\frac{1}{2}u_t^\top\Sigma^{-1}u_t + v_t^\top\Sigma^{-1}u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V) + \sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t + 2v_t^\top\Sigma^{-1}u_t\right)\right] \end{aligned} \tag{3.15}$$

Substitute Equation 3.5 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:

$$\begin{aligned} F(S,\mathbb{P},x_0,\lambda) &\le \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[S(V) + \sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t + 2v_t^\top\Sigma^{-1}u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T) + \sum_{t=1}^{T-1}q(x_t) + \sum_{t=0}^{T-1}\frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t + \beta_t^\top u_t\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\phi(x_T) + \sum_{t=0}^{T-1}\left(q(x_t) + \frac{\lambda}{2}\left(u_t^\top\Sigma^{-1}u_t + \beta_t^\top u_t\right)\right)\right] \end{aligned} \tag{3.16}$$

where the RHS is exactly the same as the objective function in Equation 3.11. 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 3.11, the controlled distribution $\mathbb{Q}_{U,\Sigma}$ should be pushed to make the RHS of Equation 3.8 as close as possible to the lower bound of free energy.

3.2.4. 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:

$$q^*(V) = \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}S(V)\right)p(V) \tag{3.17}$$

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

$$\begin{aligned} KL(\mathbb{Q}^*||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{q^*(V)}{P(V)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{1}{\eta}\exp\left(-\frac{1}{\lambda}S(V)\right)\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[-\frac{1}{\lambda}S(V) - \log(\eta)\right] \\ &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] - \log(\eta) \end{aligned} \tag{3.18}$$

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

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] + \lambda KL(\mathbb{Q}^*||\mathbb{P}) &= \mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] - \mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] - \lambda\log(\eta) \\ &= -\lambda\log(\eta) \\ &= F(S,\mathbb{P},x_0,\lambda) \end{aligned} \tag{3.19}$$

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

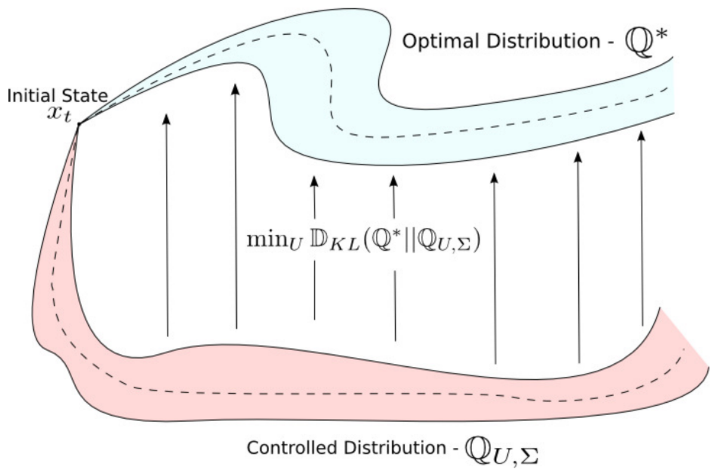
Optimal Distribution - $\mathbb{Q}^*$
Initial State
x_t
$\min_U \mathbb{D}_{KL}(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma})$
Controlled Distribution - $\mathbb{Q}_{U,\Sigma}$

Figure 3.1: Pushing the controlled distribution to the optimal one [26]

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 3.1. If $\mathbb{Q}_{U,\Sigma}$ is aligned with $\mathbb{Q}^*$, then sampling from $\mathbb{Q}_{U,\Sigma}$ will result in low-cost trajectories.

3.2.5. 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:

$$U^* = \underset{U\in\mathcal{U}}{\operatorname{argmin}}\, KL\left(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}\right) \tag{3.20}$$

where the objective KL-Divergence can be formulated as:

$$\begin{aligned} KL\left(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}\right) &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{q^*(V)}{q(V|U,\Sigma)}\right)\right] \\ &= \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q^*(V)\right)\right] - \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right] \end{aligned} \tag{3.21}$$

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:

$$\begin{aligned}
U^* &= \underset{U\in\mathcal{U}}{\text{argmax}}\,\mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right] \\
&= \underset{U\in\mathcal{U}}{\text{argmax}}\,\mathbb{E}_{\mathbb{Q}^*}\left[-\log\left(\left((2\pi)^m|\Sigma|\right)^{\frac{1}{2}}\right) - \frac{1}{2}\sum_{t=0}^{T-1}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)\right] \\
&= \underset{U\in\mathcal{U}}{\text{argmin}}\,\mathbb{E}_{\mathbb{Q}^*}\left[\frac{1}{2}\sum_{t=0}^{T-1}(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)\right] \\
&= \underset{U\in\mathcal{U}}{\text{argmin}}\,\mathbb{E}_{\mathbb{Q}^*}\left[\frac{1}{2}\sum_{t=0}^{T-1}\left(v_t^\top\Sigma^{-1}v_t + u_t^\top\Sigma^{-1}u_t - 2u_t^\top\Sigma^{-1}v_t\right)\right] \\
&= \underset{U\in\mathcal{U}}{\text{argmin}}\,\mathbb{E}_{\mathbb{Q}^*}\left[\sum_{t=0}^{T-1}\left(\frac{1}{2}u_t^\top\Sigma^{-1}u_t - u_t^\top\Sigma^{-1}v_t\right)\right] \\
&= \underset{U\in\mathcal{U}}{\text{argmin}}\left(\sum_{t=0}^{T-1}\frac{1}{2}u_t^\top\Sigma^{-1}u_t - \sum_{t=0}^{T-1}u_t^\top\Sigma^{-1}\mathbb{E}_{\mathbb{Q}^*}[v_t]\right)
\end{aligned} \tag{3.22}$$

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:

$$\begin{aligned}
&u_t^* = \mathbb{E}_{\mathbb{Q}^*}[v_t] = \int q^*(V)v_t\,\mathrm{d}V, \\
&\forall t \in \{0,1,\ldots,T-1\}
\end{aligned} \tag{3.23}$$

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.

3.3. 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 two common importance sampling techniques. Subsequently, some practical issues need to be addressed with respect to how to shift the trajectory costs, handle the control constraints, and smooth the solution. Finally, the complete information-theoretic MPPI algorithm is presented.

3.3.1. Iterative Importance Sampling

To estimate the optimal control solution in Equation 3.23, importance sampling technique could be used to generate the required samples [104]. Given an importance sampling control sequence U, the optimal control solution can be rewritten as:

$$\begin{aligned} u_t^* &= \int q^*(V) v_t \, \mathrm{d}V = \int \frac{q^*(V)}{q(V|U,\Sigma)} q(V|U,\Sigma) v_t \, \mathrm{d}V \\ &= \int \omega(V) q(V|U,\Sigma) v_t \, \mathrm{d}V \\ &= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\omega(V) v_t \right] \end{aligned} \tag{3.24}$$

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:

$$\begin{aligned} \omega(V) &= \frac{q^*(V)}{p(V)} \frac{p(V)}{q(V|U,\Sigma)} \\ &= \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(V) \right) \frac{p(V)}{q(V|U,\Sigma)} \end{aligned} \tag{3.25}$$

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 3.13, the second term can be simplified to:

$$\begin{aligned} \frac{p(V)}{q(V|U,\Sigma)} &= \frac{\exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} v_t^\top \Sigma^{-1} v_t \right)}{\exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} (v_t - u_t)^\top \Sigma^{-1} (v_t - u_t) \right)} \\ &= \exp\left(\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t - v_t^\top \Sigma^{-1} u_t \right) \right) \end{aligned} \tag{3.26}$$

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

$$\begin{aligned} \frac{p(V)}{q(V|U,\Sigma)} &= \exp\left(\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t - (u_t + \epsilon_t)^\top \Sigma^{-1} u_t \right) \right) \\ &= \exp\left(\sum_{t=0}^{T-1} \left(-\frac{1}{2} u_t^\top \Sigma^{-1} u_t - \epsilon_t^\top \Sigma^{-1} u_t \right) \right) \\ &= \exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} \left(u_t^\top \Sigma^{-1} u_t + 2\epsilon_t^\top \Sigma^{-1} u_t \right) \right) \end{aligned} \tag{3.27}$$

Substituting it into Equation 3.25 finally yields:

$$\omega(V) = \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\left(S(V) + \frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t + 2\epsilon_t^\top\Sigma^{-1}u_t\right)\right)\right) \tag{3.28}$$

$$u_t' = \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[\omega(V)v_t\right], \forall t \in \{0, 1, \ldots, T-1\} \tag{3.29}$$

$$u^* = u_0' \tag{3.30}$$

$$\eta = \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\right] = \int \exp\left(-\frac{1}{\lambda}S(V)\right)p(V)\mathrm{d}V \tag{3.31}$$

Equation 3.28 describes the importance sampling weight between the current induced distribution and the optimal distribution. Equation 3.29 updates the importance sampling sequence. Equation 3.30 is the first element of the importance sampling sequence which can be used to approximate the stochastic optimal control problem. Equation 3.31 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.

3.3.2. Covariance Variable Importance Sampling

As shown in Equation 3.26, the variance in the controlled distribution $Q_{U,\Sigma}$ is set to the natural variance of the system Σ. However, using a higher sampling variance than the natural variance can be beneficial. Thus the likelihood ratio becomes:

$$\frac{p(V)}{q(V|U,\Sigma)} = \frac{\exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}v_t^\top\Sigma^{-1}v_t\right)}{\exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}(v_t-u_t)^\top(\nu\Sigma)^{-1}(v_t-u_t)\right)} \tag{3.32}$$

where $\nu \geq 1$ is the multiplier of the magnitude of the variance. And it can be simplified as:

$$\begin{aligned}\frac{p(V)}{q(V|U,\Sigma)} &= \exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}\left(v_t^\top\Sigma^{-1}v_t - v_t^\top(\nu\Sigma)^{-1}v_t + 2v_t^\top(\nu\Sigma)^{-1}u_t - u_t^\top(\nu\Sigma)^{-1}u_t\right)\right)\\ &= \exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}\left((1-\nu^{-1})v_t^\top\Sigma^{-1}v_t + \nu^{-1}\left(2v_t^\top\Sigma^{-1}u_t - u_t^\top\Sigma^{-1}u_t\right)\right)\right)\end{aligned} \tag{3.33}$$

Rewriting $v_t = u_t + \epsilon_t$, the term inside summation becomes:

$$\begin{aligned}&(1-\nu^{-1})(u_t^\top\Sigma^{-1}u_t + 2u_t^\top\Sigma^{-1}\epsilon_t + \epsilon_t^\top\Sigma^{-1}\epsilon_t) + \nu^{-1}(2u_t^\top\Sigma^{-1}u_t + 2\epsilon_t^\top\Sigma^{-1}u_t - u_t^\top\Sigma^{-1}u_t)\\ &= (1-\nu^{-1})(u_t^\top\Sigma^{-1}u_t + 2u_t^\top\Sigma^{-1}\epsilon_t + \epsilon_t^\top\Sigma^{-1}\epsilon_t) + \nu^{-1}(u_t^\top\Sigma^{-1}u_t + 2\epsilon_t^\top\Sigma^{-1}u_t)\\ &= u_t^\top\Sigma^{-1}u_t + 2u_t^\top\Sigma^{-1}\epsilon_t + (1-\nu^{-1})\epsilon_t^\top\Sigma^{-1}\epsilon_t\end{aligned} \tag{3.34}$$

Thus, the likelihood ratio can be simplified as:

$$\frac{p(V)}{q(V|U,\Sigma)} = \exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t + 2u_t^\top\Sigma^{-1}\epsilon_t + (1-\nu^{-1})\epsilon_t^\top\Sigma^{-1}\epsilon_t\right)\right) \tag{3.35}$$

Then the weighting term is:

$$\begin{aligned}\omega(V) &= \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}S(V)\right)\frac{p(V)}{q(V|U,\Sigma)}\\ &= \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\left(S(V) + \frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t + 2u_t^\top\Sigma^{-1}\epsilon_t + (1-\nu^{-1})\epsilon_t^\top\Sigma^{-1}\epsilon_t\right)\right)\right)\end{aligned} \tag{3.36}$$

which is the same as the standard importance sampling weight in Equation 3.28 except for an extra term penalizing a large value of ϵ. The multiplier ν can also be time-varying, which makes it possible to adapt ν based on a given task online. However, special care should be paid since decreasing the covariance too much can lead to task failure in an MPC setting [105].

3.3.3. Practical Issues

The iterative importance sampling skeleton in Equation 3.28 - Equation 3.31 and the weighting equation for covariance variable importance sampling in Equation 3.36 form the basis of sampling based control methodology. But there are some practical issues to address before introducing the whole algorithm:

1) Shifting the range of the trajectory costs

The first issue is due to the numerical sensitiveness in the negative exponentiation term in the importance sampling weight, which could lead to zero weighting term or overflow errors if the trajectory cost is too high or not bounded from below. Thus, the range of the costs can be shifted in a way that the costs are bounded by the minimum cost so that the best trajectory, whose cost is the minimum cost, has a value of 0 for the negative exponentiation, while at least one trajectory has a value larger than 0. This is first done by expanding the normalizing term η in Equation 3.31 so that the importance weighting term becomes:

$$\frac{\exp\left(-\frac{1}{\lambda}\left(S(V)+\frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t+2u_t^\top\Sigma^{-1}\epsilon_t\right)\right)\right)}{\int\exp\left(-\frac{1}{\lambda}\left(S(V)+\frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t+2u_t^\top\Sigma^{-1}\epsilon_t\right)\right)\right)\mathrm{d}V}$$

Then define positive constant ρ as the minimum cost, which is chosen as the minimum sampled cost in the Monte-Carlo approximation, and multiply the above equation by:

$$1=\frac{\exp\left(\frac{1}{\lambda}\rho\right)}{\exp\left(\frac{1}{\lambda}\rho\right)}$$

which results in:

$$\omega(V)=\frac{1}{\eta'}\exp\left(-\frac{1}{\lambda}\left(S(V)-\rho+\frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t+2u_t^\top\Sigma^{-1}\epsilon_t\right)\right)\right) \tag{3.37}$$

$$\eta'=\int\exp\left(-\frac{1}{\lambda}\left(S(V)-\rho+\frac{\lambda}{2}\sum_{t=0}^{T-1}\left(u_t^\top\Sigma^{-1}u_t+2u_t^\top\Sigma^{-1}\epsilon_t\right)\right)\right)\mathrm{d}V \tag{3.38}$$

which prevents the numerical overflow or underflow by restricting the weighting term in the range [0, 1]. And η' is in the range $[1, K]$, where K is the number of samples. It can be used to indicate the performance of the algorithm. If η' is close to 1, λ or the sampling variance might be too high, then only the best trajectory is selected and all the others are discarded. If η' is close to K, λ or the sampling variance might be too low, an unweighted average could be taken.

2) Handling control constraints

The second issue is to consider the control constraints, such as the actuator limits, in the stochastic dynamical system. It can be addressed by pushing the control constraints into the system dynamics. Thus, the function of system dynamics in Equation 3.1 becomes:

$$x_{t+1}=F(x_t,g(v_t)) \tag{3.39}$$

where $g(v_t)$ is a clamping function that puts control input in an allowable region. Since computing the importance sampling weight and updating the sampling sequence does not require computing gradients or linearizing the dynamics, adding this non-linear and non-smooth component is admissible. In addition, it has no effect on the algorithm or the convergence of the importance sampling sequence.

3) Smoothing the solution

The third issue is due to the stochastic sampling procedure leading to chattering in the control sequence. It can be addressed by smoothing the output control sequence. Recall that the objective in the information-theoretic framework (Equation 3.22) can be written as:

$$u_t^*=\underset{u_t}{\operatorname{argmin}}\,\mathbb{E}_{\mathbb{Q}^*}\left[(v_t-u_t)^\top\Sigma^{-1}(v_t-u_t)\right]$$

To smooth the control sequence, an effective way is to fit local polynomial approximations at every timestep so that $u_t = a_0 + a_1 t + a_2 t^2 + \ldots + a_k t^k = A\boldsymbol{t}$, where $A = (a_0, a_1, \ldots, a_k)$ and $\boldsymbol{t} = (1, t, t^2, \ldots, t^k)$. Substituting it into the above optimization scheme yields to find the optimal coefficients at time step j:

$$A_j^* = \underset{A}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}^*}\left[\sum_{t=(j-k)}^{j+k} (v_t - A\boldsymbol{t})^\top \Sigma^{-1} (v_t - A\boldsymbol{t})\right]$$

which is equivalent to the minimization:

$$A_j^* = \underset{A}{\operatorname{argmin}} \sum_{t=(j-k)}^{j+k} (\mathbb{E}_{\mathbb{Q}^*}[v_t] - A\boldsymbol{t})^\top \Sigma^{-1} (\mathbb{E}_{\mathbb{Q}^*}[v_t] - A\boldsymbol{t}) \tag{3.40}$$

A naive method to compute the control is to compute $\mathbb{E}_{\mathbb{Q}^*}[v_t]$ using a Monte-Carlo approximation, and solve for each A_j to compute the smoothed control inputs. Another method that achieves the same result is to use a Savitsky-Galoy filter which uses a set of convolution coefficients to fit local polynomial smoothing. Thus, the smoothed control sequence can be obtained by first computing $U' = \mathbb{E}_{\mathbb{Q}^*}[V]$ and passing U' to the convolutional filter.

4) GPU-Based Parallel Sampling

Another issue is sampling a large number of trajectories fast enough for online optimization. This can be achieved by sampling in parallel on a GPU with Nvidia's CUDA architecture, which is also convenient for storing and retrieving a large amount of data.

3.3.4. Algorithm

The full algorithm of information-theoretic MPPI [26] 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 3 Model Predictive Path Integral Control [26]

```
 1: Given:
 2: $F, g$: Dynamics and clamping function;
 3: $K, T$: Number of samples and timesteps;
 4: $U$: Initial control sequence;
 5: $\Sigma, \nu, \lambda$: Parameters
 6: $\phi, q$: 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 3.1
16:             $x$ += $F(x, g(v_t))\Delta t$;                                     ▷ Equation 3.39
17:             // Update state cost and importance sampling weights
18:             $\tilde{S}_k$ += $q(x) + \frac{\lambda}{2}\left(u_t^\top \Sigma^{-1} u_t + 2u_t^\top \Sigma^{-1}\epsilon_t + (1 - \nu^{-1})\epsilon_t^\top \Sigma^{-1}\epsilon_t\right)$;   ▷ Equation 3.36
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 3.38
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 3.37
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
```

3.4. Discussion

We summarize the presented motion planning solutions in the following table, underlining their pros and cons.

Table 3.1: Comparison between motion planning methods

Motion Planning	Methods	Pros	Cons
Sampling -based	PRMs, RRTs	1. rely on the collision checking module without the need of constructing an explicit representation of the environment	1. the generated paths are normally jerky to the goal, which requires a local planner or controller to smooth and shorten the computed trajectories
		2. mostly probabilistic complete, which means if there exists a solution, the algorithm will find the solution as the number of the samples increases to infinity	2. considerable computational effort is spent in creating and connecting samples in the configuration space that might not be relevant to the task.
		3. some are asymptotically optimal, which will provably converge to an optimal solution with minimal computational and memory requirements	
Optimization -based	MPC, CHOMP, STOMP, TrajOpt	1. the algorithm that plans in continuous space is optimal and complete if the algorithm is not prone to local minima	1. obstacles, state constraints, and control constraints often require solving a non-convex, discontinuous constrained optimization problem, which is hard to solve directly and exactly
		2. the algorithm is computationally efficient due to the advances of the optimization solver	
Learning -based	IL, RL	1. user-friendly	1. IL requires an expert demonstration to initialize the learning process
		2. suitable to mimic human behaviours	2. requires a large amount of data from these interactions
			3. faces sim-to-real issues

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. As mentioned in section 3.1, 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.

Some recent works have also focused on using sampling-based MPC or predictive models to achieve manipulation tasks. Stochastic Tensor Optimization for Robot Motion (STORM) [106] was proposed to produce smooth (low-jerk) trajectories for full joint space control while taking into account the task and joint space constraints. It focused on the development of novel action sampling techniques, smooth interpolation, and cost-function design and has achieved a control frequency of 125 Hz. A hybrid learning approach [107] was proposed by combining model-based predictive models and experience-based model-free policies. The switching between the modalities works as follows: a learned model will be used to predict how well a model-free policy will behave, if it is uncertain it will use the model-based actions, otherwise, it will fall back on the experience-based policy. MuJoCo MPC (MJPC) [108] was provided as an open-source software framework for predictive control, including implementations of standard algorithms such as iLQG, Gradient Descent, and a sampling-based method.

3.5. Summary

This chapter covers the typical motion planning methods for high-dimensional robots, including sampling-based, optimization-based, and learning-based methods. Their advantages and drawbacks are discussed. A recently proposed sampling-based optimization framework MPPI draws our attention due to its advantages in dealing with complex constraints and dynamics. The key idea of MPPI is to sample control sequences from a given distribution and forward simulated 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. Finally, the discussions are made by comparing the latest works in sampling-based MPC and predictive sampling.

4 Integrated Task and Motion Planning

The previous chapters consider the problem of task planning and motion planning separately. However, this is ineffective when the robot is conducting complex tasks and actions in dynamic environments. This leads to formulating the problem of Task and Motion Planning (TAMP) in an integrated way, which will be the main focus of this chapter. This chapter aims at covering the typical TAMP approaches for high-dimensional systems. In addition, special focus will be paid to how to incorporate reactive behaviors within TAMP framework against disturbances and uncertain events. Finally, discussions will be made about the comparison between the state-of-art TAMP approaches and what could be suitable benchmarks in our case study.

4.1. TAMP

As introduced in the previous chapters, the task planning community, from a high-level perspective, focuses on solving planning problems in discrete domains and has developed many effective, domain-independent search algorithms to generate a sequence of actions for the robot to achieve some goals; research in motion planning, from a low-level perspective, focuses on generating collision-free trajectories in configuration space.

However, in order for the robot to complete complex tasks in unstructured human environments, considering pre-computed action skeleton and collision-free trajectories is not enough, planning needs to take into account a larger space including the objects it will manipulate and interact with humans or objects in a reactive way. In addition, the entire world space is highly under-actuated, since many degrees of freedom cannot be changed in arbitrary directions and the position of an object can only be changed by actuating the robot to move over and touch it. Thus, Task and Motion Planning (TAMP) seeks to combine AI approaches to task planning and robotic approaches to motion planning and it has made great progress in solving complex manipulation problems in the last decade [30].

TAMP is characterized as a hybrid discrete-continuous search problem or a Hybrid Constraint Satisfaction Problem (H-CSP), which involves searching a combination of discrete mode types (such as which action to make and which object to manipulate), continuous mode parameters (such as the grasp or contact position of the objects), and continuous motion parameters connecting configurations in the same or different modes. The general TAMP problem is complex and computationally difficult in theory [109, 110] and requires algorithmic sophistication.

In fact, the modal structure of the configuration space of the world has been observed by researchers in [111–113]: the legitimate changes to the world are in different modes, each of which is a feasible submanifold of the full space. In order to switch to a new mode, the robot has to move to the configuration which lies at the intersection of two submanifolds. For example, as illustrated in Figure 4.1, two specific feasible space $\mathcal{F}_\sigma$ and $\mathcal{F}_{\sigma'}$ correspond to modes σ and σ'. In order for the robot at configuration q to switch to a new mode σ', a path has to be made from q to the transition configuration q', where the transition configuration $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ has to satisfy the constraints of two modes. And the key challenge here is to find these configurations that lie in the intersection of the constraints of two modes. It might be intuitive to sample the configurations and reject those that do not meet both constraints, but it is time-consuming and inefficient. An alternative approach is to parameterize the constraints as $f_C(q) = 0$, then use gradient descent to project the invalid configuration to a valid one. This projection technique can be used to enable the robot to perform tasks such as opening doors [114] and keeping the orientation of target objects [115].

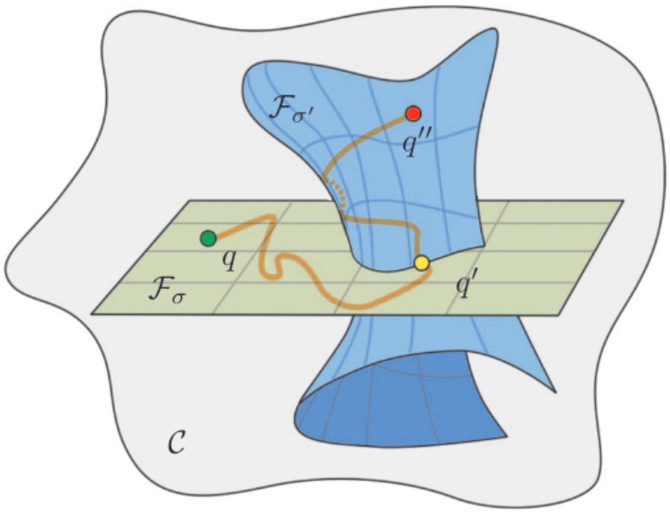

Figure 4.1: Moving between modes [112]

4.2. Task-based TAMP

4.2.1. Formalism

The majority of the TAMP problems are represented in a task-planning formulation, as shown in [30]. Similar to the classical task planning formalism as shown in subsection 2.2.1, this TAMP formalism also leverages *factoring* to decompose the state space into the Cartesian product of some subspaces via different state variables. In addition, it also represents *action* in terms of *precondition* and *effect*. The major differences lie in two aspects. One is the use of continuous action parameters, the other is the use of a new type of clause *constraint* within an action. The constraints are normally used to introduce geometric information into high-level actions, including motion trajectory, collision-checking, and kinematics.

We provide an example to articulate this TAMP formalism in the pick-and-place domain. The example contains three movable objects A, B, and C. The system is represented by the variables (atRob, holding, at[A], at[B], and at[C]), where holding is a discrete variable taking values from (None, A, B, C) and the others are variables taking continuous values. For instance, q and q' represent the robot configurations, τ represents the robot trajectory, p, p^A, p^B, p^C are the poses of the movable objects, g is a grasp pose for an object. The action moveF means the robot moves when the gripper is free and the action moveH[obj] means the robot moves when holding obj, both of which describe the motion within modes from q to q'. The actions pick[obj] and place[obj] represent switching between modes. The action templates are described as:

$$
\begin{aligned}
&\text{moveF}(q, \tau, q', p^A, \ldots, p^C)\\
&\textbf{constraint: } \text{Motion}(q, \tau, q'), \text{CFreeA}(p^A, \tau), \ldots, \text{CFreeC}(p^C, \tau)\\
&\textbf{precondition: } \text{holding = None, atRob = } q, \text{at[A]} = p^A, \ldots, \text{at[C]} = p^C\\
&\textbf{effect: } \text{atRob} \leftarrow q'\\
&\text{moveH[obj]}(g, q, \tau, q', p^A, \ldots, p^C)\\
&\textbf{constraint: } \text{Motion}(q, \tau, q'), \text{CFreeX[obj]}(p^X, g, \tau), X \in \{A, B, C\} \text{ and } X \neq \text{obj}\\
&\textbf{precondition: } \text{holding = obj, atRob = } q, \text{at[obj] = } q, \text{at[X]} = p^X, X \in \{A, B, C\} \text{ and } X \neq \text{obj}\\
&\textbf{effect: } \text{atRob} \leftarrow q'\\
&\text{pick[obj]}(q, p, g)\\
&\textbf{constraint: } \text{Stable[obj]}(p), \text{Grasp[obj]}(g), \text{Kin[obj]}(q, p, g)\\
&\textbf{precondition: } \text{holding = None, atRob = } q, \text{at[obj] = } p\\
&\textbf{effect: } \text{holding} \leftarrow \text{obj}, \text{at[obj]} \leftarrow g\\
&\text{place[obj]}(q, p, g)\\
&\textbf{constraint: } \text{Stable[obj]}(p), \text{Grasp[obj]}(g), \text{Kin[obj]}(q, p, g)\\
&\textbf{precondition: } \text{holding = obj, atRob = } q, \text{at[obj] = } g\\
&\textbf{effect: } \text{holding} \leftarrow \text{None}, \text{at[obj]} \leftarrow p
\end{aligned}
\tag{4.1}
$$

As shown in the above action templates, each action can have a set of constraints. For instance, the Motion(q, τ, q') constraint specifies the start and end configuration of trajectory τ, which satisfies $\tau(0) = q, \tau(1) = q'$; the collision-free constraint CFreeX[obj](p^X, g, τ) means it is collision-free to execute trajectory τ if X is at p^X and obj is held at pose g; similarly, the collision-free constraint CFreeA(p^A, g, τ) means it is collision-free to execute trajectory τ if A is at p^A; the constraint Stable[obj](p) means pose p is a stable placement for obj; the constraint Grasp[obj](g) means g is a stable relative pose relative to the hand frame to grasp obj; the constraint Kin[obj](q, p, g) means that if the robot in in configuration q and holding obj at relative pose g, then obj will be in configuration p.

Consider the TAMP problem with the initial state $s_0 = \{\text{atRob=}\mathbf{q_0}, \text{at[A]=}\mathbf{p_0}, \text{holding=None}\}$ and the goal state $s_g = \{\text{at[A]=}\mathbf{p_g}, \text{holding=None}\}$, where $\mathbf{p_0}, \mathbf{q_0}, \mathbf{p_g}$ are specified real-valued constants. Then one possible plan skeleton can be found:

$$
\begin{aligned}
\pi = [&\text{moveF}(\mathbf{q_0}, \tau_1, q_1), \text{pick[A]}(q1, \mathbf{p_0}, g_1),\\
&\text{moveH[A]}(q_1, \tau_2, q_2), \text{place[A]}(q_2, \mathbf{p_g}, g_1)]
\end{aligned}
\tag{4.2}
$$

where $\tau_1, g_1, q_1, \tau_2, q_2$ are the free parameters that need to be selected. Figure 4.2 illustrates the plan skeleton in a graph, where the gray nodes represent constant values and colored nodes represent variable values. Each row represents the state variables and their possible values. Each column corresponds to a state and each action that shows state changes within two columns is listed at the top. The rectangular nodes represent constraints and each constraint is connected to the constrained variables.

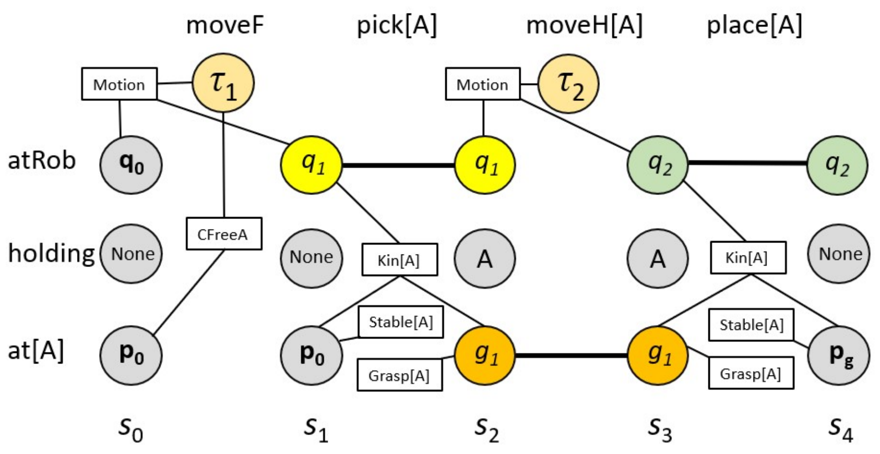

Figure 4.2: TAMP plan skeleton

There are different strategies integrating the discrete and continuous components of the TAMP problems: *sequence before satisfy*, where the action sequence is iteratively selected before trying to solve the constrained motion planning problem; *satisfy before sequence*, where trajectories are iteratively found for individual specific constraints before assembling these actions into complete plans; and *interleaved*, where planned action and satisfying constraints are added incrementally. The flowchart of the first two strategies is shown in Figure 1.3.

4.2.2. Related Works

The pioneering aSyMov planner [116, 117] combines a heuristic-search task planner Metric-FF [118] with lazily expanded roadmaps based on Move3D [119]. The main contribution of the aSyMov planner is to build the relationship between a search for configurations and a symbolic position, which is done via a developed instantiation process that builds *accessibility list* and propagates task and geometric constraints in the search process. At each step, the planner selects possible actions and computes costs and heuristics based on the heuristic estimator. An action that will bring the symbolic state to the is more likely to be chosen. The symbolic levels guide the search to solve a relaxed version of the problem in which all paths are considered valid. The roadmap expands when valid actions are added. The planner also has to balance the time between finding a plan with the level of knowledge already obtained and investing more time to obtain a deeper knowledge of the topology of the manipulated configuration subspaces. The search process terminates until valid geometric configurations are found that also satisfy the plan at geometric and symbolic levels. The found plan and trajectories will be further improved by parallelizing the symbolic plan and coordination of trajectories in a post-processing phase.

Hierarchical Planning in the Now (HPN) [120] deals with the interleaved task and motion planning in a hierarchical architecture. The architecture is in a top-down fashion that tries to limit the length of the plans that need to be constructed and thereby speeds up the search time. It operates on continuous geometric representations and does not require a priori discretization of the state or action space.

Srivastava et al. [121] proposed an extensible planner-independent interface layer that interfaces off-the-shelf task planners and motion planners. The interface can translate geometric information, such as trajectories or errors, normally as preconditions to the task planner in terms of logical predicates. In order to avoid computing expensive precondition properties, the interface assumes the properties to be true for the task planner. If there is no feasible plan found for the default values, the task planner will invoke the motion planner to check the geometric precondition. And the task planner is repeated to find a new plan after the new geometric precondition is updated.

FFrob [122, 123] is an integrated task and motion planer using a search where the heuristic takes into account the geometric and kinematics information. The heuristic extends the ideas from the FastForward symbolic task planner, which integrates reachability in both the robot configuration space and the symbolic state space. In addition, in order to figure out the effects of each symbolic action, FFRob maintains a state representation that contains both true literals and a data structure to capture the geometric information to any true literal. The data structure is referred to *Conditional Reachability Graph* (CRG), which is similar to a PRM structure and allows answering multiple motion planning reachability queries conditioned on the placements and grasps of objects, lazily computing answers on demand and caching results for future queries.

PDDLStream [124] is a sampling-based task and motion planner which also allows action planners to validate preconditions of the symbolic actions considering the geometric constraints. It extends the standard PDDL by incorporating *streams*, which contain a generic, declarative specification for inverse kinematic solvers, collision checkers, and motion planners. Each symbolic action contains one or more *streams*. Each *stream* contains a *conditional generator* that generates output values based on input values, such as generating new robot configurations that satisfy a kinematic constraint with potential poses and grasp positions. And the implementation of the *conditional generator* is treated as a black box for the user's convenience. However, PDDLStream requires an explicit specification of geometric constraints within the symbolic description.

TMKit [125] is the first general-purpose probabilistically complete task and motion planning framework. The task domain is created in the syntax of PDDL and the motion domain is represented by a new motion scene graph that supports direct task and motion translation and efficient updates. The task and motion domains are related via the domain semantics which defines a function to map from a scene graph to a discrete state in the task planner and a function to map from a discrete action to a computed corresponding motion plan. TMKit can be incorporated with the algorithm of [126, 127], which employs an incremental, constraint-based task planner in the task layer and some sampling-based planners from the Open Motion Planning Library (OMPL) [128] in the motion layer.

Lagriffoul et al. [129, 130] identified *geometric backtracking* as one of the major problems in TAMP, which is described as a planner has to reconsider geometric choices for previous actions in order to satisfy the precondition of the current action. This may lead to intractable computation due to a large number of geometric configurations even with a coarse discretization. To alleviate this issue, an interface is introduced to generate a set of approximate linear constraints from the symbolic actions and the geometric level. Linear programming techniques are then used to compute *intervals* to reduce the space of geometric configurations. The *intervals* are narrowed by the constraints to contain all possible solutions to the problem. This approach is efficient for geometrically constrained tasks such as stacking multiple objects.

4.3. Optimization-based TAMP

4.3.1. Formalism

Most of the existing TAMP approaches are dominated by task-based formalism, which build upon a well-defined task planning problem and often rely on predicates to represent geometric preconditions of symbolic actions. Different strategies are used to invoke or integrate with motion planners. Some of them try to prune infeasible decision branches as quickly as possible, while others try to *lazily* post-pone expensive geometric computation until a solution to the plan skeleton is found. Most of these approaches focus on improving the computing performance when scaling to a large number of objects and a large number of actions. In short, *task-based TAMP tries to pull geometry into logic representations.*

However, most of the task-based TAMP approaches focus on finding feasible action sequences instead of the optimal ones. Even though it is difficult to reduce the H-CSPs to a constrained mathematical program due to the inclusion of discrete parameters, continuous parameters, and non-convex constraints, some progress has been made to formulate the problem as a continuous mathematical program that can be solved locally optimally using constrained optimization.

Logic-Geometric Programming (LGP) [131] has been proposed by Marc Toussaint to *pull logic into mathematical programming* for sequential manipulation, where the optimization is directly on the geometric level and discrete logic is to control the constraints in different modes. To consider the robot and all objects, let $x : [0, T] \to \mathcal{X}$ be a trajectory in the configuration space $\mathcal{X}$, and the initial configuration $x(0)$ is given. In addition, the problem is based on a first-order logic $\mathcal{L}$, an initial state $s_0 \in \mathcal{L}$, a sequence of symbolic transitions $s_k = succ(a_k, s_{k-1})$ that transform the current logic state s_{k-1} to the next state s_k given action a_k, and $K + 1$ time points $t_k \in [0, T], k = 0, 1, \dots, K$ with $t_0 = 0, t_K = T$. Then the LGP problem can be formulated as:

$$\begin{aligned} \min_{x, a_{1:K}, s_{1:K}, t_{1:K}} \quad & \int_0^T c(x(t), \dot{x}(t), \ddot{x}(t))\mathrm{d}t + \psi(x(T)) \\ \text{s.t.} \quad & s_k = succ(a_k, s_{k-1}), \forall k = 1 : K \\ & s_K \models \mathfrak{g} \\ & h_{switch}(x(t_k)|a_k, s_{k-1}) = 0, \forall k = 1 : K \\ & g_{switch}(x(t_k)|a_k, s_{k-1}) \leq 0, \forall k = 1 : K \\ & h_{path}(x(t), \dot{x}(t)|s_{k-1}) = 0, t \in [t_{k-1}, t_k], \forall k = 1 : K \\ & g_{path}(x(t), \dot{x}(t)|s_{k-1}) \leq 0, t \in [t_{k-1}, t_k], \forall k = 1 : K \end{aligned} \tag{4.3}$$

where the objective function is an optimal control on a trajectory, with running cost c and terminal cost ψ on the final configuration. The constraints h_{switch} and g_{switch} are equality and inequality constraints that relate to switching between modes. The constraints h_{path} and g_{path} are equality and inequality constraints that relate to geometric motion information between modes during time point t_{k-1} and t_k. The final logic state s_K is also required to be a symbolic goal within $\mathfrak{g}$. This formalism allows the objective function to encode the final geometric configuration. This is different from the case in task-based TAMP, where the final logic goal is normally specified as an objective.

4.3.2. Related Works

In order to solve this question in a feasible way, Marc Toussaint proposed three steps to make geometric reasoning inform symbolic search [131].

- *Optimizing over the effective end space*. The *effective end space* is defined as all configurations $x(T)$ that can be reached conditioning on a given action sequence $a_{1:K}$. Optimizing the final configuration over the effective end space can be considered a robot-pose optimization problem and can be used as a heuristic to guide the symbolic search. In addition, this approximation is efficient to solve since it does not consider the running costs and constraints between modes, which will be considered in the next two steps.
- *Optimizing over all switch configurations*. At this step, the switch configurations $\{x(t) : t = t_0, t_1, \dots, t_K\}$ are optimized conditioning on the given action sequence $a_{1:K}$. The optimization also accounts for long-term dependencies, which mean the dependencies of trajectories across different modes, based on the K-Order Motion Optimization framework (KOMO) [132].
- *Optimizing over the full path*. To optimize the full trajectory $x : [0, T] \to \mathcal{X}$ conditioning on the given action sequence $a_{1:K}$, this is done by taking into account 20 time steps between the switch configurations with an interpolation of the optimized switch configurations. The accelerations of all switch configurations are assumed to be zero to allow independent optimization during each mode period.

To obtain the symbolic action sequences, LGP uses basic Monte Carlo Tree Search (MCTS), which is less efficient than the existing TAMP approaches. In each iteration of MCTS the action sequences $a_{1:K}$ are obtained, then optimize the final configuration with respect to terminal objective $\psi(x_T)$. Then step 2 optimization is performed to find feasible switch configurations and finally step 3 optimization is performed to find path x and solution $(x, a_{1:K}, s_{1:K})$. LGP has been applied and extended to the cases of placing multiple blocks and boards, tool use and dynamic interaction [133], multi-robot cooperative manipulation [134], and long-horizon architectural construction [135].

The approach of Lozano-Perez and Kaelbling [136] also formulates the TAMP problem in a constraint-based way, but it reduces the problem to a CSP instead of a continuous mathematical program. And it also focuses on finding a feasible solution instead of an optimal one.

A Satisfiability Modulo Convex programming (SMC) [137, 138] framework has been proposed to integrate discrete Boolean actions and convex constraints at the same time. The idea is similar to pulling logic and constraints into geometric reasoning. It has been applied to the case of robot motion planning for reach-avoid problems, spacecraft docking mission control, and secure state estimation. But it has not yet applied directly to the domain of TAMP with respect to combining different actions and motions.

4.4. Reactive TAMP

Aforementioned 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 retail store 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. Some related works can be shown below.

Robust Logical-Dynamical Systems (RLDS) [139] has been proposed for task execution which supports fallback or recovery behaviors. Compared with Behavior Tree and Finite State Machine, this system does not specify the internal structure of a task and preconditions of each state, instead, it contains a list of operators with specified preconditions, run conditions, and effects. These operators will lead the system towards the convergence to the goal. It can be combined with real-time motion generator Riemannian Motion Policies (RMPs) [140] or RL policies. The system has shown reactivity to disturbances in a case study of household manipulation tasks.

Stouraitis et al. [141] proposed an online hybrid motion planning framework for dyadic collaborative manipulation. The problem requires a human and a robot co-manipulate an object and it assumes the robot can predict the partner's intentions as task goals. The framework can generate hybrid action plans online which react to the dyadic goal changes. This is done by a bilevel optimization formulation combining graph search method A* and trajectory optimization, where the discrete properties include the contact states of end-effectors, and geometric properties include contact locations, forces, and timings.

Li et al. [142] proposed a reactive TAMP framework to generate reactive and cost-effective plans for manipulation tasks. The framework consists of three layers, including a high-level planning layer, middle-level execution layer, and low-level control layer. The high-level planning layer can generate a complete and semi-optimal task plan by using Linear Temporal Logic (LTL) and approximated motion costs. The middle-layer execution layer is a behavior tree that conditionally activates subtrees to achieve reactiveness based on the high-level plans. The low-level control layer generates control commands via the primitive controller. The framework is tested in a robot to perform a pick-and-place task of multiple objects which are subject to environmental changes such as object relocation, addition, and removal of objects and regions.

Timing-Optimal Sequence-of-Constraints MPC (SECMPC) [143] has been proposed to execute a sequence of constraints that are subject to perturbations during execution. The problem assumes the sequence of constraints (waypoint constraints and running constraints) are given beforehand by TAMP approaches and it focuses on the whole horizon problem to obtain better timing and waypoint consistency, unlike traditional short receding horizon MPC. Then the control problem is divided into three subproblems including determining sequential waypoints, the timing of waypoints, and the receding horizon path which fulfill collision constraints and path constraints. These subproblems are solved online at each MPC iteration. The algorithm also takes into account the long-term interdependencies of constraints, which means current reactiveness may have to consider the constraints in the future. It also includes a backtracking phase when running costs are missed. The algorithm was demonstrated in the case of a pushing-with-stick scenario with human perturbations and replacing target objects.

4.5. Discussion

The state-of-the-art TAMP solutions are summarized in the following table and their properties are also highlighted.

Table 4.1: Comparison Between TAMP Methods

Methods	Formalism	Task Planner	Motion Planner	Combining Strategies	Reactive	Tasks
aSyMov [116, 117]	Task-based	Heuristic-search Metric-FF	PRM	Sequence first	No	Pick-and-place
HPN [120]	Task-bsaed	Hierarchical tree	RRT	Interleaved	No	Houshold tasks (pick-and-place, clean, swap, wash)
Srivastava et al. [121]	Task-based	Off-the-shelf task planner	Off-the-shelf motion planner	Sequence first	No	Pick object from cluttered table, lay out a table
FFrob [122, 123]	Task-based	FastForward planner	PRM	Satisfaction first	No	Pick object from cluttered table
PDDLStream [124]	Task-based	Extended PDDL with streams	PRM	Sequence first	No	Houshold tasks (pick, place, stack, push, press, pour, scoop)
TMKit [125]	Task-based	Constraint-based task planner	OMPL	Sequence first	No	Pick, place, set table
Lagriffoul et al. [129, 130]	Optimization -based	Geometric-backtracking algorithm	RRT	Sequence first	No	Pick-and-place
LGP [131]	Optimization -based	Monte Carlo Tree Search	K-Order Motion Optimization Framework	Sequence first	No	Placing multiple blocks and boards, tool use and dynamic interaction [133], multi-robot cooperative manipulation [134], and long-horizon architectural construction [135].
SMC [137, 138]	Optimization -based	None	SAT solver	None	No	Motion planning for reach-avoid problems,
RLDS [139]	Task-based	RLDS	RMPs	Sequence first	Yes	Household manipulation
Stouraitis et al. [141]	Optimiaztion -based	Graph search method A*	Trajectory optimization	Interleaved	Yes	Dyadic collaborative manipulation
Li et al. [142]	Task-based	Linear Temporal Logic, Behaviour Tree	Primitive controller	Sequence first	Yes	Pick-and-place task of multiple objects
SECMPC [143]	Optimization -based	TAMP planner	Sequential MPC	Sequence first	Yes	Pushing-with-stick

In the context of mobile manipulation of this literature, we consider reactive methods necessary because the real-world environment in a retail store is dynamically changing where there are disturbances and unexpected events interfering with the robots. Therefore, robots should be equipped with reactive behaviors to operate safely and efficiently in dynamic and uncertain environments. Among these reactive TAMP methods, there are some good alternatives but they lack in some aspects. For instance, RLDS [139] and Li's work [142] are not open-source, SECMPC [143] and STORM [106] are open-source and might be suitable candidates when comparing the low-level action execution, but they are not capable of reasoning and selecting different actions online. Thus, we propose in the next chapter a possible approach to tackle the problem of mobile manipulation in an uncertain dynamic environment.

4.6. Summary

This chapter builds the connection between task planning and motion planning and aims to address the problem of *downward refinement* in an integrated way. The chapter covers the two main formalisms of TAMP, which are formulated in a task-planning style and an optimization framework. We review the state-of-the-art works in these two categories and highlight their properties. We pay special attention to the TAMP approaches that can achieve reactive behaviors because this is an essential property in our research case. Finally, the discussions have been made by comparing the related works and pinpointing their limitations, namely less focus on generating versatile manipulation skills and achieving reactive properties from both high and low levels, which help to propose a possible approach to fill in the research gap.

5 Research Proposal

The goal of this proposal is to design a control scheme that can combine different actions and online action selections to achieve reactive behaviors, making the motion planning problem contact rich. In addition, the time schedule of the thesis is provided.

5.1. Method

The proposed Reactive TAMP scheme is shown in the figure below. It aims to address the problem which has been mentioned in section 1.1. We discuss in the following the main blocks of the scheme, namely the *Goals, Task Planner, Motion Planner, Interface, and Use Cases.*

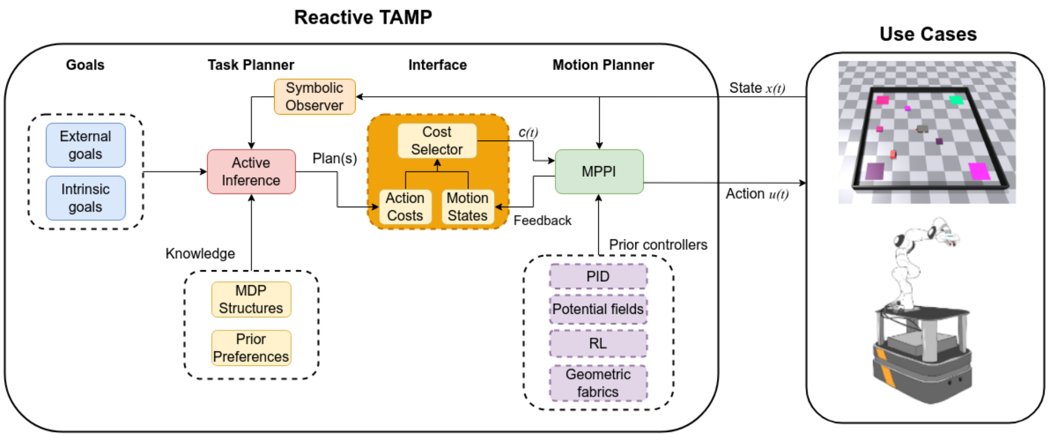

Figure 5.1: Reactive TAMP scheme

5.1.1. Goals

The TAMP scheme takes user input as goals. The goals can be categorized into two aspects:

- External goals: this type of goal considers encoding the desired state or behavior we want the robot to achieve. A behavior tree can be used to provide global reactivity when conducting long-horizon tasks.
- Internal goals: the robot is also equipped with some internal goals that require certain states to be maintained at certain levels. For instance, we can consider the robot to be associated with an internal battery level, if the battery level is too low, the robot cannot execute the current action and thus needs go to recharge.

5.1.2. Task Planner

The task planner is responsible for making online action selections based on the encoded goals and current states. Since we want to apply our framework to the case of a retail store, we pay special attention to the task planner that can generate reactive plans against disturbances and unexpected events. After the extensive literature study, active inference draws our attention due to its appealing aspects shown as follows:

- A partial skeleton containing desired states is provided, not action with parameters.
- The actions are flexibly adapted at run-time.
- It is flexible in encoding parameterized habits and preferences.

The task planner will be fed with some knowledge first. This includes MDP structures and some parameters, such as prior preferences over actions. At run time, after receiving the states from the symbolic observer, the task planner will generate reactive plans and send them to the interface. It should be noticed that the task planner can generate a single plan if it is certain about the current situation and solutions, otherwise it will generate alternative plans. The alternatives will later be evaluated by the cost selector and executed in the low-level motion planner.

5.1.3. Motion Planner

The motion planner aims to generate efficient, contact-rich manipulation motions in dynamic environments. This remains an issue for high-dimensional systems such as mobile manipulation. MPPI, a sampling-MPC method, was proposed to overcome the drawbacks of typical MPC approaches by allowing flexible parallel sampling. The advantages of MPPI are shown as follows:

- Sampling different actions in parallel on a physics engine has great potential for mobile robots.
- It is flexible in dealing with constraints, such as obstacle, state, and control constraints and cost functions, which do not require linear and quadratic approximations and can even treat discontinuous cases.
- The advanced use of GPU enables fast control loop frequency in parallel sampling and allows efficient collision checking.

It should be mentioned that although the motion planner could be combined with several prior controllers as shown in the scheme, it is not a contribution of my thesis.

5.1.4. Interface

In order to achieve simultaneous planning from both high level and low level, the interface plays a key role in integrating the actions and the motions. The core component of the interface is a cost selector that will schedule the costs of different actions by considering the action costs and motion states. The interface has to play three roles when connecting the task planner and the motion planner:

- Translate symbolic costs of different actions to the motion planner via the cost selector. This indicates which action to choose among the alternative symbolic plans.
- Send low-level motion feedback to the cost selector to choose alternative plans if the current action fails to execute.
- Enable the robot to achieve both long-term and short-term cognitive aspects of the task, which is driven by a coherent cost function within the interface.

5.1.5. Use Cases

The use cases can be a mobile robot and a mobile manipulation. The system consider conducting experiments both in simulation and on real robots. For the simulation work, we consider choosing the physics simulator Isaac Gym [144]. The advantages of Issac Gym are shown as follows:

- Isaac Gym provides efficient kinematic and dynamic models of the robots. This is very important in the domain of TAMP, which requires rapid updates to kinematic and dynamic equations when the robot changes actions.
- Isaac Gym allows efficient parallel sampling of multiple environments.
- Isaac Gym allows utilization of GPU, which will fasten the computing.

In addition, we also consider some benchmarks to compare in our use cases.

- Sense-plan-act: this is a typical hierarchical robot control framework that relies on downward refinement, which always assumes every correct high-level plan can be correctly executed. This may be problematic when the robot operates in dynamic, uncertain environments and the high-level actions will rely on the low-level motion states.
- Our TAMP + heuristic observers: this approach relies on some heuristic observers to determine when and where the robot should complete which actions. However, this may face issues when the actions are hard to determine from simple heuristics, such as pull and push.
- STORM [106]: this is an open-source framework that uses sampling-based MPC to achieve fast, low-jerk trajectories for manipulators. It is a suitable benchmark to compare low-level action generation, but it is not capable of reasoning and completing complex tasks online.

5.2. Time Schedule

In the time schedule the first week starts on January 1, 2023. It records the tasks I have done in the past since last June and also schedules the thing to do for the thesis. Hopefully, the thesis will be finished and the defense will be done before the end of June.

Week	**Phase**	**Task Description**
-28	Initialization	Discuss thesis topic
-27		
-26	Literature survey	Review task planning techniques
-25		
-24		
-23		
-22		Review active inference
-21		
-20		
-19		Review motion planning techniques
-18		
-17		Review MPPI
-16		
-15		
-14		Review task and motion planning
-13		
-12		
-11	Simulation	Get familiar with IssacGym
-10		Inherit from open source MPPI file to move the robot from A to B
-9		Build and make elegant code structure
-8		Add visualization to motion trajectories
-7		Add collision avoidance
-6		Implement pull action for mobile robots
-5		AMR group presentation
-4	Writing	Write literature survey
-3		
-2		Hand in survey report to daily supervisors

-1		Discuss, modify, and polish
0	Christmas	Holiday
1	Simulation	Integrate with task planner
2		Design task-motion interface
3		Integrate different actions (push, pull) for mobile robots
4		
5		
6		AMR group presentation / Colloquia
7		
8		Implement different actions (pick, place, pull, push) for mobile manipulation
9		
10		Integrate different actions (pick, place, pull, push) for mobile manipulation
11		
12		
13		Compare with benchmark
14		
15	Writing	Gather results of simulation
16		Analyze and write
17	Real-world experiments	Test in mobile robots
18		Test in mobile manipulation
19		
20	Writing	Analyze and write
21		Write
22		Discuss with daily supervisors / AMR group presentation
23		Modify and polish
24		Hand in thesis report
25	Prepare defense	Prepare presentation and practice
26		Final presentation and defense

6 Conclusion

This literature survey reviews the state-of-the-art task planning, motion planning, and integrated task and motion planning approaches for mobile manipulation and special focus is paid to *how to generate versatile reactive behaviors and efficient motion planning techniques in multi-task, contact-rich, uncertain, and dynamic environments.*

The main question can be divided into three sub-questions. The first sub-question is: *how does the mobile manipulator generate high-level reactive plans to cope with uncertainties and unexpected events?* The answer has been given in chapter 2, providing an overview of the task planning problems and different approaches. There are three main approaches to achieving high-level robot autonomous behaviors: the programming-based, learning-based, and planning-based approaches. Planning-based methods suit our needs on account of the fact that they are generally model-based and can provide a guarantee of the solution. Apart from that, they are also highly interpretable and show great generalization power compared with learning-based methods. Among these formal methods, the free-energy principle and active inference have been introduced as strong candidates in order to achieve reactive behaviors in real time. 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 flexibility in encoding parameterized habits and preferences.

Once the high-level action sequence is decided, it would be natural to ask *how can a mobile manipulator achieve efficient contact-rich manipulation motions in dynamic environments?* To answer this, the first step is to get an overview of different approaches to motion planning. Typical motion planning approaches can be categorized into sampling-based, optimization-based, and learning-based. After extensive research, a variety of methodologies are found and analyzed, underlining the benefits and limits of these algorithms. Among these typical methods, MPPI attracts our attention due to its flexibility in dealing with constraints and cost functions and the ability to combine prior controllers to achieve efficient sampling. In addition, the advanced use of GPU enables fast control loop frequency in parallel sampling and allows efficient collision checking. 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.

The aforementioned chapters try to consider task planning and motion planning separately, however, this may raise many issues in real-world applications where high-level actions could vary depending on the low-level motion states and changing environments. Thus, it is necessary to consider the question: *how can the task planning and motion planning be incorporated to facilitate the interdependence?* The fourth chapter provides an overview of the state-of-the-art TAMP approaches and analyzes two major formalisms. One formalism tries to pull geometry into logic representation and is represented in a task-planning style, which builds upon a well-defined task-planning problem and uses different strategies to invoke or integrate with motion planners. The other formalism, on the other hand, tries to pull logic into mathematical programming and is formulated in an optimization problem. This chapter also introduces how to design the interdependence layers and what strategies can be used to combine task level and motion levels. In addition, much focus is on the approaches of reactive TAMP that can handle disturbances and unexpected events. This property is essential in our case in a retail store. Some latest works are listed but they lack in some aspects, namely less focus on generating versatile manipulation skills and achieving reactive properties from both high level and low level.

In the research proposal, the control scheme combining active inference and MPPI for the reactive task and motion planning is proposed. The scheme could provide adaptability at the task level through online decision making, as well as low-level reactiveness by means of a parallel sampling motion planner. The key component of this framework is the design of the interdependence interface that contains a cost selector that will schedule the costs of different actions by considering the action costs and motion states. Moreover, one great potential contribution of this work lies in that the interface enables the robot to achieve both long-term and short-term cognitive aspects of the task, which is driven by a coherent cost function within the interface. The considered use cases are using a mobile robot or mobile manipulator to perform different tasks, involving actions such as pushing, pulling, picking, and placing in human-centered environments.

To conclude, this survey summarises the existing literature regarding reactive task and motion planning techniques for mobile manipulation, identifying the research gaps in the current research. Furthermore, this document aims to provide sufficient theoretical knowledge to conduct the future thesis work, namely performing a simulation and a real-world test to prove or disprove the promise of the proposed reactive TAMP scheme. Demonstrating the correctness and completeness of such a scheme would suggest a new way of achieving versatile manipulation skills in the real-world scenarios.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Grady Williams et al. “Robust Sampling Based Model Predictive Control with Sparse Objective Information.” In: *Robotics: Science and Systems*. 2018.

[28] Jintasit Pravitra et al. “L1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 7661–7666.

[29] Ji Yin et al. “Improving Model Predictive Path Integral using Covariance Steering”. In: *CoRR* abs/2109.12147 (2021). arXiv: `2109.12147`. URL: `https://arxiv.org/abs/2109.12147`.

[30] 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.

[31] Hector Geffner. “The model-based approach to autonomous behavior: A personal view”. In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010.

[32] Michael Beetz et al. “AI reasoning methods for robotics”. In: *Springer Handbook of Robotics*. Springer, 2016, pp. 329–356.

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

[34] Nils J Nilsson et al. “Shakey the robot”. In: (1984).

[35] Fahiem Bacchus and Qiang Yang. “Downward refinement and the efficiency of hierarchical problem solving”. In: *Artificial Intelligence* 71.1 (1994), pp. 43–100.

[36] Constructions Aeronautiques et al. “PDDL| The Planning Domain Definition Language”. In: *Technical Report, Tech. Rep.* (1998).

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

[38] Rina Dechter, Itay Meiri, and Judea Pearl. “Temporal constraint networks”. In: *Artificial intelligence* 49.1-3 (1991), pp. 61–95.

[39] Simone Fratini et al. “Apsi-based deliberation in goal oriented autonomous controllers”. In: *ASTRA* 11 (2011).

[40] Javier Barreiro et al. “EUROPA: A platform for AI planning, scheduling, constraint programming, and optimization”. In: *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)* (2012).

[41] Malik Ghallab and Hervé Laruelle. “Representation and Control in IxTeT, a Temporal Planner.” In: *Aips*. Vol. 1994. 1994, pp. 61–67.

[42] Michael Cashmore et al. “Robustness envelopes for temporal plans”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 7538–7545.

[43] Tom Bylander. “The computational complexity of propositional STRIPS planning”. In: *Artificial Intelligence* 69.1-2 (1994), pp. 165–204.

[44] Blai Bonet and Héctor Geffner. “Planning as heuristic search”. In: *Artificial Intelligence* 129.1-2 (2001), pp. 5–33.

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

[46] Peter E Hart, Nils J Nilsson, and Bertram Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[47] Anton Riabov, Shirin Sohrabi, and Octavian Udrea. “New algorithms for the top-k planning problem”. In: *Proceedings of the scheduling and planning applications workshop (spark) at the 24th international conference on automated planning and scheduling (icaps)*. 2014, pp. 10–16.

[48] Michael Katz et al. “A novel iterative approach to top-k planning”. In: *Twenty-Eighth International Conference on Automated Planning and Scheduling*. 2018.

[49] Shirin Sohrabi, Anton V Riabov, and Octavian Udrea. “Plan Recognition as Planning Revisited.” In: *IJCAI*. New York, NY. 2016, pp. 3258–3264.

[50] Rebecca Eifler et al. “Explaining the space of plans through plan-property dependencies”. In: (2019).

[51] Sam Toyer et al. “Action schema networks: Generalised policies with deep learning”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[52] Daniel Gnad et al. “Learning how to ground a plan–partial grounding in classical planning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 7602–7609.

[53] Randal E Bryant. “Graph-based algorithms for boolean function manipulation”. In: *Computers, IEEE Transactions on* 100.8 (1986), pp. 677–691.

[54] Alvaro Torralba. “Symbolic search and abstraction heuristics for cost-optimal planning”. In: *Universidad Carlos III de Madrid* (2015).

[55] David Speck, Robert Mattmüller, and Bernhard Nebel. “Symbolic top-k planning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 06. 2020, pp. 9967–9974.

[56] Alvaro Torralba et al. “SymBA*: A symbolic bidirectional A* planner”. In: *International Planning Competition*. 2014, pp. 105–108.

[57] David Speck, Florian Geißer, and Robert Mattmüller. “Symbolic planning with edge-valued multi-valued decision diagrams”. In: *Twenty-Eighth International Conference on Automated Planning and Scheduling*. 2018.

[58] Malte Helmert. “The fast downward planning system”. In: *Journal of Artificial Intelligence Research* 26 (2006), pp. 191–246.

[59] Karl Johan Åström. “Optimal control of Markov processes with incomplete state information”. In: *Journal of mathematical analysis and applications* 10.1 (1965), pp. 174–205.

[60] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[61] Hanna Kurniawati. “Partially observable markov decision processes and robotics”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2022), pp. 253–277.

[62] Christos H Papadimitriou and John N Tsitsiklis. “The complexity of Markov decision processes”. In: *Mathematics of operations research* 12.3 (1987), pp. 441–450.

[63] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. “Point-based value iteration: An anytime algorithm for POMDPs”. In: *Ijcai*. Vol. 3. Citeseer. 2003, pp. 1025–1032.

[64] David C Knill and Alexandre Pouget. “The Bayesian brain: the role of uncertainty in neural coding and computation”. In: *TRENDS in Neurosciences* 27.12 (2004), pp. 712–719.

[65] Kenji Doya et al. *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.

[66] Laurent Itti and Pierre Baldi. “Bayesian surprise attracts human attention”. In: *Vision research* 49.10 (2009), pp. 1295–1306.

[67] Mike Oaksford and Nick Chater. “A rational analysis of the selection task as optimal data selection.” In: *Psychological Review* 101.4 (1994), p. 608.

[68] Jonathan D Nelson. “Finding useful questions: on Bayesian diagnosticity, probability, impact, and information gain.” In: *Psychological review* 112.4 (2005), p. 979.

[69] Pedro Tsividis et al. “Information selection in noisy environments with large action spaces”. In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 36. 36. 2014.

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

[71] Alireza Soltani et al. “Neural substrates of cognitive biases during probabilistic inference”. In: *Nature communications* 7.1 (2016), pp. 1–14.

[72] Dobromir Rahnev and Rachel N Denison. “Suboptimality in perceptual decision making”. In: *Behavioral and Brain Sciences* 41 (2018).

[73] 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.

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

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

[76] 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.

[77] 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.

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

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

[80] 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.

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

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

[83] John H Reif. “Complexity of the mover’s problem and generalizations”. In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE Computer Society. 1979, pp. 421–427.

[84] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[85] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[86] Joshua Bialkowski, Sertac Karaman, and Emilio Frazzoli. “Massively parallelizing the RRT and the RRT”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3513–3518.

[87] Stefano Carpin and Enrico Pagello. “On parallel RRTs for multi-robot systems”. In: *Proc. 8th Conf. Italian Association for Artificial Intelligence*. 2002, pp. 834–841.

[88] Didier Devaurs, Thierry Siméon, and Juan Cortés. “Parallelizing RRT on large-scale distributed-memory architectures”. In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 571–579.

[89] Wen Sun, Sachin Patil, and Ron Alterovitz. “High-frequency replanning under uncertainty using parallel sampling-based motion planning”. In: *IEEE Transactions on Robotics* 31.1 (2015), pp. 104–116.

[90] Alexandros Paraschos et al. “Probabilistic movement primitives”. In: *Advances in neural information processing systems* 26 (2013).

[91] Jan Peters and Stefan Schaal. “Reinforcement learning of motor skills with policy gradients”. In: *Neural networks* 21.4 (2008), pp. 682–697.

[92] Jens Kober and Jan Peters. “Policy search for motor primitives in robotics”. In: *Advances in neural information processing systems* 21 (2008).

[93] Jens Kober et al. “Reinforcement learning to adjust parametrized motor primitives to new situations”. In: *Autonomous Robots* 33.4 (2012), pp. 361–379.

[94] Guilherme Maeda et al. “Acquiring and generalizing the embodiment mapping from human observations to robot skills”. In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 784–791.

[95] Alex X Lee et al. “Learning force-based manipulation of deformable objects from multiple demonstrations”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 177–184.

[96] Jihong Zhu et al. “Challenges and outlook in robotic manipulation of deformable objects”. In: *IEEE Robotics & Automation Magazine* 29.3 (2022), pp. 67–77.

[97] Jens Kober, J Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[98] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. “A survey on policy search for robotics”. In: *Foundations and Trends® in Robotics* 2.1–2 (2013), pp. 1–142.

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

[100] 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.

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

[102] 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.

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

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

[105] Freek Stulp and Olivier Sigaud. “Path integral policy improvement with covariance matrix adaptation”. In: *arXiv preprint arXiv:1206.4621* (2012).

[106] Mohak Bhardwaj et al. “Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 750–759.

[107] Allison Pinosky et al. “Hybrid control for combining model-based and model-free reinforcement learning”. In: *The International Journal of Robotics Research* (2022), p. 02783649221083331.

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

[109] Ashwin Deshpande, Leslie Pack Kaelbling, and Tomas Lozano-Perez. “Decidability of semi-holonomic prehensile task and motion planning”. In: *Algorithmic foundations of robotics XII*. Springer, 2020, pp. 544–559.

[110] Marilena Vendittelli, Jean-Paul Laumond, and Bud Mishra. "Decidability of robot manipulation planning: Three disks in the plane". In: *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 641–657.

[111] Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. "A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps". In: *The fifth international symposium on Robotics research*. MIT Press. 1990, pp. 453–463.

[112] Kris Hauser and Jean-Claude Latombe. "Multi-modal motion planning in non-expansive spaces". In: *The International Journal of Robotics Research* 29.7 (2010), pp. 897–915.

[113] Kris Hauser and Victor Ng-Thow-Hing. "Randomized multi-modal motion planning for a humanoid robot manipulation task". In: *The International Journal of Robotics Research* 30.6 (2011), pp. 678–698.

[114] Mitsuharu Kojima, Kei Okada, and Masayuki Inaba. "Manipulation and recognition of objects incorporating joints by a humanoid robot for daily assistive tasks". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 1564–1569.

[115] Mike Stilman. "Task constrained motion planning in robot joint space". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 3074–3081.

[116] Fabien Gravot, Stephane Cambon, and Rachid Alami. "aSyMov: a planner that deals with intricate symbolic and geometric problems". In: *Robotics Research. The Eleventh International Symposium*. Springer. 2005, pp. 100–110.

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

[118] Jörg Hoffmann. "The Metric-FF Planning System: Translating"Ignoring Delete Lists"to Numeric State Variables". In: *Journal of artificial intelligence research* 20 (2003), pp. 291–341.

[119] Thierry Simeon, J-P Laumond, and Florent Lamiraux. "Move3D: A generic platform for path planning". In: *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560)*. IEEE. 2001, pp. 25–30.

[120] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Hierarchical planning in the now". In: *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010.

[121] Siddharth Srivastava et al. "Combined task and motion planning through an extensible planner-independent interface layer". In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 639–646.

[122] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "Ffrob: An efficient heuristic for task and motion planning". In: *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 179–195.

[123] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. "Ffrob: Leveraging symbolic planning for efficient task and motion planning". In: *The International Journal of Robotics Research* 37.1 (2018), pp. 104–136.

[124] 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.

[125] Neil T Dantam, Swarat Chaudhuri, and Lydia E Kavraki. "The task-motion kit: An open source, general-purpose task and motion-planning framework". In: *IEEE Robotics & Automation Magazine* 25.3 (2018), pp. 61–70.

[126] Neil T Dantam et al. "An incremental constraint-based framework for task and motion planning". In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1134–1151.

[127] Neil T Dantam et al. "Incremental task and motion planning: A constraint-based approach." In: *Robotics: Science and systems*. Vol. 12. Ann Arbor, MI, USA. 2016, p. 00052.

[128] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. “The open motion planning library”. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[129] Fabien Lagriffoul et al. “Constraint propagation on interval bounds for dealing with geometric backtracking”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 957–964.

[130] Julien Bidot et al. “Geometric backtracking for combined task and motion planning in robotic systems”. In: *Artificial Intelligence* 247 (2017), pp. 229–265.

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

[132] Marc Toussaint. “Newton methods for k-order Markov constrained motion problems”. In: *arXiv preprint arXiv:1407.0414* (2014).

[133] Marc A Toussaint et al. “Differentiable physics and stable modes for tool-use and manipulation planning”. In: (2018).

[134] Marc Toussaint and Manuel Lopes. “Multi-bound tree search for logic-geometric programming in cooperative manipulation domains”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4044–4051.

[135] Danny Driess, Ozgur Oguz, and Marc Toussaint. “Hierarchical task and motion planning using logic-geometric programming (hlgp)”. In: *RSS Workshop on Robust Task and Motion Planning*. 2019.

[136] Tomás Lozano-Pérez and Leslie Pack Kaelbling. “A constraint-based method for solving sequential manipulation planning problems”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 3684–3691.

[137] Yasser Shoukry et al. “Scalable lazy SMT-based motion planning”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 6683–6688.

[138] Yasser Shoukry et al. “SMC: Satisfiability modulo convex programming”. In: *Proceedings of the IEEE* 106.9 (2018), pp. 1655–1679.

[139] Chris Paxton et al. “Representing robot task plans as robust logical-dynamical systems”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5588–5595.

[140] Ching-An Cheng et al. “RMPflow: A computational graph for automatic motion policy generation”. In: *International Workshop on the Algorithmic Foundations of Robotics*. Springer. 2018, pp. 441–457.

[141] Theodoros Stouraitis et al. “Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization”. In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1452–1471.

[142] 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.

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

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