TUDelft

Delft University of Technology

Exploring Active Inference and Model Predictive Path Integral Control
A Journey from Low-Level Commands to Task and Motion Planning

Pezzato, C.

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Exploring Active Inference and Model Predictive Path Integral Control

A Journey from Low-Level Commands to Task and Motion Planning

Corrado Pezzato

Exploring Active Inference and Model Predictive Path Integral Control

A Journey from Low-Level Commands to Task and Motion Planning

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof. dr. ir. T. H. J. J. van der Hagen;
Chair of the board of Doctorates
to be defended publicly on
Tuesday the 9$^{\text{th}}$ of January 2024, at 10:00 o'clock

by

Corrado PEZZATO

Master of Science in Systems and Control
Delft University of Technology, The Netherlands
born in Mirano, Italy

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr.ir. M. Wisse,	Delft University of Technology, promotor
Dr.ir. C. Hernández Corbato,	Delft University of Technology, copromotor

Independent members:

Prof.dr. M.M. de Weerdt,	Delft University of Technology
Prof.dr.hc. M. Beetz,	University of Bremen
Prof.dr. C.L. Buckley,	University of Sussex
Prof.dr. A. Wąsowski,	IT University of Copenhagen
Dr. C. Della Santina,	Delft University of Technology
Prof.dr.ir. J. Hellendoorn,	Delft University of Technology, reserve member.

An electronic version of this dissertation is available at http://repository.tudelft.nl/.

To the collective knowledge of humanity ...

Contents

Summary

I<small>N</small> an ever-evolving society, the demand for autonomous robots equipped with human-level capabilities is becoming increasingly imperative. Various factors, such as an aging population and a shortage of labor for repetitive and physically demanding tasks, have underscored the need for capable autonomous robots to assist us in our daily activities. However, despite the recent advancements in robotics, the field still faces significant challenges in delivering on its promises of developing general-purpose robots with human-level capabilities for everyday tasks. *This thesis aims to develop control algorithms at different levels of abstraction to achieve more robust, adaptive, and reactive robot behavior for long-term tasks in dynamic environments.*

Since our ultimate goal is to achieve human-level performance, a natural starting point is to investigate theories of human intelligence and how they can be applied to real robots, such as mobile manipulators. In this regard, one prominent theory is *Active Inference*, a popular and influential concept that can explain a wide range of cognitive functions, from motor control to high-level decision-making. Active Inference was developed based on the free-energy principle providing an explanation for embodied perception-action loops. While the free-energy principle and Active Inference have garnered significant attention among neuroscientists, their application to robotics remains largely unexplored, presenting an exciting avenue for research in this thesis. At the same time, it is also important to recognize that we should not confine ourselves solely to theories of human intelligence and their inherent limitations. Machines and humans are built upon fundamentally different structures, which opens up possibilities for alternative approaches. Consequently, this thesis also investigates the use of *Model Predictive Path Integral Control* (MPPI), which stems from a different formulation of free-energy that is not bound to biological assumptions. By exploring the application of Active Inference to low-level robot control and task planning, as well as the utilization of MPPI for motion planning, this thesis provides advancements in robot control at different levels of abstraction.

More concretely, this thesis contributes to the following four areas: 1) Low-level adaptive and fault-tolerant control, 2) Reactive high-level decision making, 3) Contact-rich motion planning, and 4) Reactive task and motion planning (TAMP).

Low-level control The research that resulted in this dissertation started with the exploration of Active Inference for low-level control of robot manipulators. In this initial phase, we proposed the first applications of this neuroscientific theory for adaptive [109] and fault-tolerant control [107]. These initial works highlighted the advantages and limitations of Active Inference for classical control. One of the core ideas in Active Inference is to bias the current belief toward a desired goal in the generative model, implicitly modeling actions. While this results in an elegant

control law that couples state estimation and control, it also leads to a biased state estimate that can cause false positives in fault-tolerant control. To overcome this we introduced the **key idea** of explicitly modeling control actions in the generative model. This led to the unbiased Active Inference scheme for classical and fault-tolerant control [6, 10] described in Chapters 3 and 4.

Reactive high-level decision making   Active Inference literature proposes both a continuous and a discrete version of the framework, for low-level control and high-level decision making respectively. The discrete version was proposed in the neuroscientific community as a unified framework to solve the exploitation-exploration dilemma by acting to minimize the free-energy. Probabilistic beliefs about the state of the world are built through Bayesian inference, and a finite horizon plan is selected to maximize the evidence for a model that is biased toward the agent's preferences. However, the use of this formulation was limited to simplified simulation scenarios with fixed preferences and relied on fundamental assumptions such as instantaneous actions without preconditions. To bring discrete Active Inference to real-world robotics, we introduced the **key idea** of planning desired states to perform a task through a Behavior Tree [96], to guide the online action selection process with Active Inference. This combination of offline and online planning led to a computationally tractable algorithm for the online synthesis of reactive robot behavior, as explained in Chapter 5.

Contact-rich motion planning   Robots operating in real environments and interacting with the surroundings necessarily need to take into account physical interaction with objects. The Active Inference approaches developed in Chapters 3 and 4 did not allow for straightforward integration of contact models for low-level behavior, while Chapter 5 considered pre-defined routines when interacting with the world. Thus, we shifted our focus to Model Predictive Path Integral Control, as a method to gracefully handle discontinuous dynamic models. However, the problem of defining a model for robot interaction still remained. Thus, Chapter 6 introduced the **key idea** of using a GPU-parallelizable physics simulator as the dynamical model for MPPI. This led to a new way of performing sampling-based model predictive control where no explicit dynamic model is required from a user [110]. This enabled us to effortlessly solve complex tasks such as non-prehensile manipulation and whole-body prehensile manipulation with high degrees-of-freedom (DOFs) robots.

Reactive task and motion planning   This dissertation concludes with a final contributing chapter combining the advances of the methods explained above. The **key idea** is to schedule costs to be minimized by MPPI through Active Inference. Active Inference is extended to plan alternative action plans, to be sampled in parallel by MPPI. By doing so, we obtained a system that can blend together different strategies to achieve the same goal, without requiring user-defined switching heuristics. Effectively merging various robot capabilities led to switching-free behaviors, at the same time allowing a robot to deal with contact-rich tasks and adjust online

the high-level plans. As explained in Chapter 7, this idea results in a reactive task and motion planning framework with a high degree of autonomy [147].

The algorithms presented in this thesis have been evaluated both from a theoretical and empirical perspective, performing in-depth mathematical analysis as well as simulations and real experiments. Through the ideas presented in this dissertation, we hope to propel the field of robotics forward and pave the way for the new generation of researchers working toward the widespread adoption of autonomous robots in our society.

Samenvatting

I<small>N</small> een voortdurend veranderende samenleving groeit de vraag naar autonome robots met menselijke capaciteiten gestaag. Diverse factoren, zoals de vergrijzing van de bevolking en een tekort aan arbeidskrachten voor herhalende en fysiek veeleisende taken, benadrukken de noodzaak van bekwame autonome robots die ons kunnen bijstaan in onze dagelijkse bezigheden. Ondanks de voortgang in de robotica, wordt het veld nog steeds geconfronteerd met aanzienlijke uitdagingen bij het vervullen van de belofte om robots te ontwikkelen die algemene taken met menselijke capaciteiten kunnen uitvoeren. *Dit proefschrift is gefocused op het ontwikkelen van besturingsalgoritmen op verschillende abstractieniveaus, met als doel robuuster, aanpasbaarder en responsiever gedrag van robots te bereiken voor langetermijntaken in dynamische omgevingen.*

Aangezien ons ultieme doel is om prestaties op menselijk niveau te bereiken, is het natuurlijk om te beginnen met het onderzoeken van theorieën over menselijke intelligentie en hoe deze kunnen worden toegepast op praktische robots, zoals mobiele manipulatoren. In dit opzicht is *Active Inference* een prominente theorie, het is een populair en invloedrijk concept dat een breed scala aan cognitieve functies kan verklaren, van motorische controle tot besluitvorming op hoog niveau. Actieve Inferentie is ontwikkeld op basis van het free-energy principle en biedt een verklaring voor de interacties tussen waarneming en actie binnen een lichaam. Hoewel het free-energy principle en de Active Inference veel aandacht hebben gekregen onder neurowetenschappers, blijft hun toepassing op robotica grotendeels onontgonnen, wat een opwindende onderzoeksrichting in dit proefschrift oplevert. Tegelijkertijd is het ook belangrijk om te erkennen dat we ons niet uitsluitend moeten beperken tot theorieën over menselijke intelligentie en hun inherente beperkingen. Machines en mensen zijn gebouwd op fundamenteel verschillende structuren, wat mogelijkheden opent voor alternatieve benaderingen. Daarom onderzoekt dit proefschrift ook het gebruik van *Model Predictive Path Integral Control* (MPPI), dat voortkomt uit een andere formulering van free-energy die niet gebonden is aan biologische aannames. Door diepgaand in te gaan op de toepassing van Actieve Inferentie in robotbesturing en laag-niveau taakplanning, en door het gebruik van MPPI voor bewegingsplanning te verkennen, draagt dit proefschrift bij aan de vooruitgang in robotbesturing op verschillende abstractieniveaus.

Meer concreet draagt dit proefschrift bij aan de volgende vier gebieden: 1) Adaptieve en fouttolerante controle op laag niveau, 2) Reactieve besluitvorming op hoog niveau, 3) Contactrijke bewegingsplanning, en 4) Reactieve taak en bewegingsplanning (TAMP).

Controle op laag Het onderzoek dat heeft geleid tot dit proefschrift begon met de verkenning van Active Inference voor de besturing op laag niveau van robotma-

nipulatoren. In deze eerste fase hebben we de eerste toepassingen van deze neuro-wetenschappelijke theorie voorgesteld voor adaptieve [109] en fouttolerante controle [107]. Deze eerste werken benadrukten de voordelen maar ook de beperkingen van Active Inference voor klassieke controlemethoden. Een van de kernideeën van Active Inference is om de huidige overtuiging in de richting van een gewenst doel in het generatieve model te sturen, waarbij impliciet acties worden gemodelleerd. Hoewel dit resulteert in een elegante controlewet die de schatting van de huidige status en de controle hiervan combineert, leidt het ook tot een vertekende schatting die kan leiden tot valse positieven in fouttolerante controle. Om dit aan te pakken, hebben we het **essentiële idee** geïntroduceerd om de controleacties expliciet te modelleren in het genratieve model. Dit leidde tot het onbevooroordeelde Active Inference-schema voor klassieke en fouttolerante controle [6, 10], beschreven in Chapters 3 and 4.

Reactieve besluitvorming op hoog niveau   Binnen de Active Inference litera-tuur wordt zowel een continue als een discrete versie van het raamwerk voorgesteld, respectievelijk voor controle op laag niveau en besluitvorming op hoog niveau. De discrete versie werd in de neurowetenschappelijke gemeenschap voorgesteld als een verenigd raamwerk om het dilemma van exploitatie en exploratie op te lossen door de free-energy te minimaliseren. Probabilistische overtuigingen over de toestand van de wereld worden opgebouwd via Bayesiaanse redernering, en er wordt een plan met een eindige horizon geselecteerd om het bewijs voor een model dat de voorkeu-ren van de actor weerspiegelt, te maximaliseren. Het gebruik van deze formulering was echter beperkt tot vereenvoudigde simulatiescenario's met vaste voorkeuren en berustte op fundamentele aannames zoals onmiddellijke acties zonder voorafgaande voorwaarden. Om discrete Active Inference te introduceren in de echte wereld van robotica, hebben we het **essentiële idee** geïntroduceerd om gewenste toestanden te plannen voor de uitvoering van een taak via een gedragsboom [96], om zo het proces van online actieselectie te verbinden met Active Inference. Deze combinatie van offline en online planning leidde tot een computationeel hanteerbaar algoritme voor de online synthese van reactief robotgedrag, zoals uitgelegd in Chapter 5.

Contactrijke bewegingsplanning   Robots die in echte omgevingen opereren en interactie hebben met de omgeving, moeten noodzakelijkerwijs rekening houden met fysieke interactie met objecten. De benaderingen gebaseerd op Active Inference, ont-wikkeld in Chapters 3 and 4, lieten geen naadloze integratie van contactmodellen toe voor laag-niveau gedrag, terwijl Chapter 5 vooraf gedefinieerde routines in overwe-ging nam tijdens de interactie met de wereld. Daarom hebben we onze focus verlegd naar Model Predictive Path Integral Control, als een methode om op elegante wijze met discontinue dynamische modellen om te gaan. Het probleem van het definiëren van een model voor robotinteractie bleef echter bestaan. Zo introduceerde Chapter 6 het **essentiële idee** van het gebruik van een GPU-paralleliseerbare fysica-simulator als het dynamische model voor MPPI. Dit leidde tot een nieuwe manier om op steek-proeven gebaseerde voorspellende controle van modellen uit te voeren, waarbij geen expliciet dynamisch model vereist is van een gebruiker [110]. Dit stelde ons in

staat om moeiteloos complexe taken op te lossen, zoals niet-grijpende manipulatie en grijpmanipulatie van het hele lichaam met robots met een hoge vrijheidsgraad (DOF).

Reactieve taak- en bewegingsplanning   Dit proefschrift sluit af met een laatste hoofdstuk dat de voortgang van de eerder uiteengezette methoden combineert. Het **essentiële idee** is om te plannen op een manier waarbij de kosten worden geminimaliseerd door middel van Model Predictive Path Integral Control (MPPI), geïntegreerd met Active Inference. Active Inference wordt uitgebreid om alternatieve actieplannen te genereren, die parallel worden bemonsterd door MPPI. Door dit te doen, hebben we een systeem verkregen dat verschillende strategieën kan combineren om hetzelfde doel te bereiken, zonder dat door de gebruiker gedefinieerde schakelheuristieken nodig zijn. Deze effectieve combinatie van diverse robotcapaciteiten resulteert in naadloos en schakelvrij gedrag, waardoor de robot contactrijke taken kan uitvoeren en tegelijkertijd online hoog-niveau plannen kan aanpassen. Zoals uitgelegd in Chapter 7 resulteert dit idee in een algemeen reactief raamwerk voor taak- en bewegingsplanning met een hoge mate van autonomie [147].

De algoritmen die in dit proefschrift worden gepresenteerd zijn zowel vanuit een theoretisch als empirisch perspectief geëvalueerd, waarbij diepgaande wiskundige analyses zijn uitgevoerd, evenals simulaties en echte experimenten. Met de ideeën die in dit proefschrift worden gepresenteerd hopen we het veld van de robotica vooruit te helpen en de weg vrij te maken voor de nieuwe generatie onderzoekers die werken aan de brede acceptatie van autonome robots in onze samenleving.

1

Introduction

*T*HIS *chapter serves as an introductory overview of the thesis, providing insights into the motivations and challenges behind this work. Particularly, in the following we outline the roadmap that guided the development of the main chapters, emphasizing the core contributions made. Additionally, we present an overview of the document's structure, facilitating readers in navigating through the research presented.*

The hallmark of a good scientist is asking questions that no one has asked before

Karl J. Friston

1

1.1. Motivations

The integration of autonomous robots into human environments that are capable of performing human-level tasks has emerged as a significant area of research, holding tremendous potential for transformative impact. In this thesis, we aim to advance the capabilities of such robots in terms of adaptive motor control and high-level decision-making. These aspects are essential for successfully navigating and interacting within environments designed for humans.

Humans possess remarkable cognitive and physical abilities that enable them to interact effortlessly with their surroundings, exhibiting complex behaviors and achieving goals efficiently. Understanding how humans accomplish various tasks is a fundamental question that has intrigued researchers for decades. Exploring the intricacies of human behavior not only contributes to our understanding of ourselves but also offers valuable insights for developing autonomous robots with similar abilities.

One conceptual framework that tries to shed light on human behavior is the theory of *Active Inference*, which is based on the notion of biological free-energy. According to the free-energy principle, biological systems like humans have to minimize free-energy to survive and behave adaptively in their environments. Derived from this idea, Active Inference has been proposed to explain and describe action, perception, and learning in humans. Investigating this concept allows us to delve deeper into the underlying principles governing human behavior and to seek parallels between biological systems and artificial systems, such as robots.

While the study of biological theories such as Active Inference provides valuable insights for mimicking human behavior, it is crucial to keep in mind that robots are built fundamentally differently than humans. Since robots do not possess biological constraints, it is important also to explore non-biologically plausible theories for autonomous behavior. During the development of this thesis, we came across a theory called *Model Predictive Path Integral Control* (MPPI). Interestingly, there exists an information-theoretic definition of MPPI that makes use of a non-biological formulation of free-energy for optimal control. By studying, adapting, and blending Active Inference and MPPI, this thesis aims to investigate novel robot control strategies at different levels of abstraction. Driven by curiosity, we leverage the benefits and address some of the limitations of both Active Inference and MPPI, aiming to achieve human-level capabilities in autonomous robots.

1.1.1. Overview of the problem

This thesis focuses on the operation of robots in environments primarily designed for humans, specifically within retail scenarios. Our goal is to deploy robots for various general activities in everyday supermarkets, such as stocking shelves or retrieving online orders. In Figure 1.1, an example image showcases one of the robots employed in this thesis, situated within a testing supermarket environment.

Retail stores present unique challenges compared to traditional robotics industrial settings, primarily due to the need for robots to coexist within the same workspace as employees and customers. This poses significant challenges across various levels of a robot's control architecture. Regarding **low-level control**, the

Figure 1.1: Meet Albert, the mobile manipulator in our mock-up store. This robot, among others, will be utilized throughout this dissertation to showcase the novel algorithms proposed.

robot must navigate the environment safely, potentially incorporating compliant control modes and fault-tolerant strategies to recover or fail-safe. Moreover, collision avoidance becomes more complex as it involves not only static obstacles like shelves but also humans whose behavior may be hard to predict.

To reliably retrieve a wide variety of products from tightly packed shelves, autonomous robots should perform **contact-rich motion planning**. Object manipulation while considering contact behaviors and interactions between objects in a retail scenario is a challenging problem that often requires specialized control algorithms. Additionally, as tasks extend over time, such as stocking a shelf or fulfilling an online order, numerous issues can arise during execution within these dynamic environments. The robot must automatically recover and seamlessly resume task execution through adaptive **high-level decision-making** and possibly **reactive task and motion planning** (TAMP).

Finally, fast control loops are required to enable the robot to swiftly adapt to the ever-changing retail environment while performing a task. With this context in mind, the following section provides an overview of the approaches and key concepts developed in this thesis to address the aforementioned challenges. In particular, we describe the roadmap that led to this thesis, from low-level control to task and motion planning.

1.1.2. Research roadmap

Low-level control

Motivated by the necessity for adaptive, compliant, and fault-tolerant low-level control in robot manipulators, the initial phase of this PhD research sought solutions in the theory of Active Inference. At the time of writing, Active Inference appeared promising in addressing these challenges, capturing our attention. We then ventured to apply this theoretical neuroscientific idea to a real robot, introducing the Active Inference controller (AIC) for the first time [109]. Through this endeavor, we discovered both the potential and limitations of Active Inference in adaptive control

for manipulators with unknown dynamics and fault tolerance [107]. Consequently, subsequent research, presented comprehensively in Chapter 3 and Chapter 4, focused on addressing false positives in fault-tolerant control with Active Inference and overcoming theoretical limitations of standard Active Inference in classical control. These efforts culminated in the development of an unbiased adaptive scheme known as the unbiased Active Inference controller (u-AIC), which eliminated state estimation biases and implicit modeling of control actions found in classical AIC.

High-level decision making

After achieving compliant and adaptive low-level control with u-AIC to drive a robot manipulator to a specific configuration, the question arose: How can a mobile manipulator effectively utilize this skill within a retail store? To attain robot autonomy, high-level decision-making capabilities became paramount. Our focus turned to advancing high-level reactive action selection to address runtime variability in retail environments. In this context, Behavior Trees (BTs) recently emerged as an intuitive framework for robustly and reactively executing offline action plans [96, 32]. However, their reactivity is constrained by the contingencies considered during offline planning. In unpredictable retail environments, encoding every possible recovery action for a task failure became impractical. A desirable high-level planning algorithm, in this context, should contain runtime planning capabilities. However, in long-term tasks, online long-horizon planning might be computationally costly. Additionally, re-planning the whole task continuously might be a waste of resources. For this reason, we proposed a different approach that combines offline and online planning. Specifically, we proposed to plan offline a sequence of states instead of actions in a BT, and subsequently utilize an online planner to transition from the current state to the next, thereby reducing computational complexity. By doing so, one can obtain a robot capable of following predefined routines, while at the same time adapting locally to unforeseen events through online planning. This concept forms the core idea presented in Chapter 5, where BTs are seamlessly blended with Active Inference. By encoding the desired state sequence within the BTs and relying on Active Inference for state estimation and action selection, we achieve a tractable and real-time online planning approach.

Contact-rich motion planning

While the approach described earlier provided significant adaptation capabilities at a high level, it still relied on pre-defined skills scheduled by Active Inference. This results in behaviors reminiscent of switching systems and requires numerous heuristics. For instance, for a simple pick action, one would need to pre-define object-specific pre and post-grasp poses that might depend on the object's location in a supermarket or define handcrafted strategies for retrieving products from the shelves. This made us focus on leveraging object geometries and physical properties more effectively to achieve smoother behaviors without relying on extensive heuristics. To do so, the research shifted to contact-rich motion planning. To tackle this problem, we utilized the theory of Model Predictive Path Integral Control [142, 143], which, like Active Inference, draws from the concept of free-energy but under

different assumptions, as explained in Chapter 2. MPPI is a method for sampling-based model predictive control capable of handling complex dynamical models with contact-induced discontinuities. Instead of relying on specialized models for planning in MPPI, we proposed using a general-purpose parallelizable physics engine, IsaacGym [85], as the dynamical model. This allowed us to achieve real-time planning capabilities over a short time horizon while considering the physical properties of objects and a reduced number of heuristics. This approach offers a powerful means of defining robot behavior in terms of cost functions. This is advantageous because one has to encode only the desired end goal as a cost to be minimized, instead of specifying (possibly many) hard-coded routines to achieve that goal. However, it is important to note that the time horizon remains limited; thus the robot is not able to tackle long-horizon tasks. A detailed explanation of this approach is provided in Chapter 6.

Reactive Task and motion planning
To be able to use the approach in Chapter 6 throughout a long-term task, we combined high-level planning with Active Inference and BTs together with MPPI. Instead of scheduling atomic skills at the high level with Active Inference, we proposed to schedule skill-specific cost functions. This seemingly simple change brought about remarkable implications for robot behavior. With this approach, we can achieve smooth transitions and seamless blending of different skills thanks to parallel sampling. In addition, we can leverage a deeper physical understanding of the world instead of relying on pre-programmed heuristics. This idea is presented in the final contributing chapter of this thesis, Chapter 7, where we tackled the combined problem of task and motion planning.

1.2. Summary of contributions
In the following, we formally summarize the main contributions resulting from the roadmap presented above. We highlight the scientific advancements in the areas of *low-level control*, *high-level decision-making*, *contact-rich motion planning*, and *task and motion planning*. In particular, this thesis contributes:

- **A new scheme for classical control with Active Inference (low-level control)**: Initially, we presented a first application of Active Inference that showed remarkable capabilities for robot arm control with unknown dynamics [109]. Additionally, we performed an initial study for fault-tolerant control with AIC [107]. Despite the interesting results, some of the problems in standard AIC are biased state estimation for classical control and false positives for fault-tolerant control. In this thesis, we formally investigate the limitations of the AIC through theoretical analysis and empirical demonstrations. Then, we mathematically formalize the unbiased Active Inference controller, u-AIC, as an improved alternative [10]. The **key idea** in the u-AIC is to explicitly model control actions in AIC in order to eliminate state bias and false positives in fault-tolerant settings [9]. We validate the effectiveness of the u-AIC through simulation and experiments on a robot manipulator. This marks the first application of the u-AIC on a real robot.

1

- **A new scheme for online decision-making based on discrete Active Inference and behavior trees** (high-level decision making): We propose a novel combination of Active Inference and BTs for reactive action planning and execution in dynamic environments [108]. This showcases how robotic tasks can be effectively formulated as a free-energy minimization problem. The **key idea** here is to use BTs to plan desired states for Active Inference instead of encoding actions as commonly done. This change enables continuous online planning with Active Inference, allowing the handling of partially observable initial states and enhancing the robustness of classical BTs when facing unforeseen contingencies. The efficacy of our approach is validated through experiments conducted on two distinct real mobile manipulators, further reinforcing the validity and practicality of our findings. Importantly, our work pioneers the application of discrete Active Inference on a real robot, marking a significant milestone in this field.

- **A new way of performing MPPI leveraging a physics simulator** (contact-rich motion planning): We introduce a novel approach for solving finite horizon optimal control problems with MPPI where the **key idea** is using a GPU-parallelizable physics simulator as the dynamical model [110]. In particular, we make use of the general-purpose simulator IsaacGym. The proposed method eliminates the need for explicit encoding of robot dynamics and object interactions, enabling effortless solving of intricate navigation and contact-rich tasks. This allows one to accommodate various objects and robots easily. Through extensive simulations, including mobile navigation with collision avoidance, non-prehensile manipulation, and whole-body control of high degrees of freedom (DOF) systems, we showcase the effectiveness of this method. Results are validated in the real world for non-prehensile manipulation with both a mobile base and a 7-DOF robotic manipulator.

- **A control architecture for reactive task and motion planning combining Active Inference and MPPI** (task and motion planning): We propose a reactive task and motion planning scheme by combining Active Inference and BTs with MPPI [147]. This scheme enables real-time adaptation of both high-level behavior and low-level robot motions while considering the physical interaction with objects. The **key idea** behind our approach is the transition from scheduling symbolic actions using Active Inference, to scheduling costs to be minimized. Since the system is now controlled by MPPI throughout the whole long-term tasks, we can sample different alternatives in parallel for achieving the current goal, blending them into one coherent behavior. By effectively merging various robot capabilities, we obtain a system that is free from abrupt switching. Results are validated both in simulation and in the real world in task and motion planning scenarios. These include pushing and pulling of objects to target goals, and stacking of cubes in the presence of dynamic obstacles with disruptive human intervention during execution.

Overall, this thesis advances the understanding of the use of Active Inference and MPPI for robot control, and it discusses the benefits and limitations of the proposed approaches toward autonomous robots.

1.3. Structure of the document

This document follows a structured progression, reflecting the evolution of the author's ideas during the doctorate[1]. A graphical representation of this structure can be seen below in Figure 1.2.
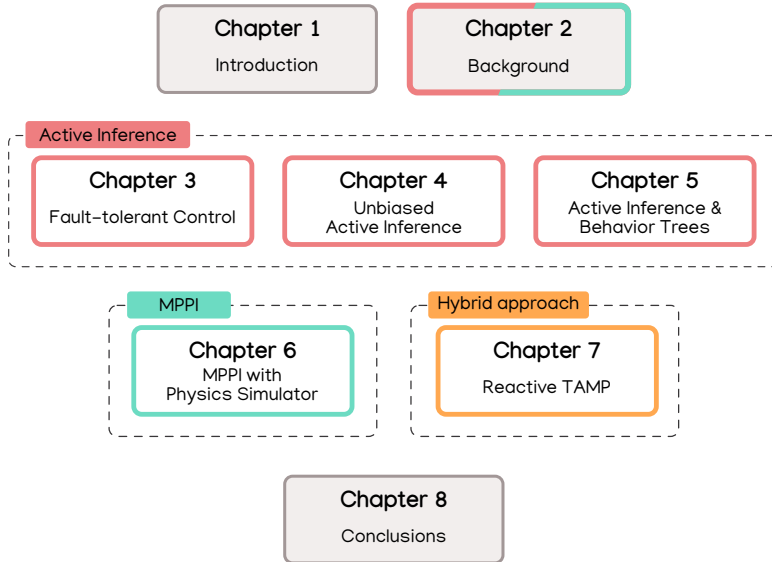
Figure 1.2: Structure of this document.

Specifically, Chapter 2 contains the background knowledge for Active Inference, MPPI, and fault-tolerant control. Subsequently, Chapter 3 presents the fault-tolerant scheme based on Active Inference, while Chapter 4 introduces the mathematical formulation of u-AIC for classical control. In Chapter 5, discrete Active Inference and behavior trees are combined for reactive action planning and execution. Moving forward, Chapter 6 explores the use of MPPI with a physics simulator for contact-rich tasks. The final contributing chapter, Chapter 7, proposes a hybrid combination of Active Inference and MPPI for reactive task and motion planning. The thesis concludes in Chapter 8, providing a summary and discussion of the methods used, along with potential future research directions.

---

[1]Parts of this document have been polished using tools such as Grammarly and ChatGPT. These tools have not been used to produce original text, but to merely improve the grammatical correctness and clarity of certain paragraphs written by the author.

2

Background

*T*HIS *chapter serves as a background on the core concepts explored in this disser-tation. Its main objective is to give readers a general understanding of Active Inference, Model Predictive Path Integral Control, and fault-tolerant control. These concepts are evolved and blended in the subsequent contributing chapters.*

2.1. Active Inference

The concept of Active Inference is rooted in the idea that the brain's cognitive functions and motor control can be understood through the lens of free-energy minimization. The notion of free-energy in Active Inference draws inspiration from living organisms' natural tendency to resist disorder, and the term itself was coined due to similarities with the thermodynamics free-energy. According to the free-energy principle, organisms ensure their survival by minimizing the surprise associated with unexpected events, quantified as "surprisal". Mathematically, surprisal is defined as the negative logarithm of the probability of certain sensory data, i.e. $-\ln p(\boldsymbol{y})$. This captures the *atypicality* of sensory data $\boldsymbol{y}$ in the environment; the greater the improbability of an event the higher the surprisal. In this context, free-energy is a weighted sum of sensory prediction errors that serves as an upper bound on the extent of sensory data atypicality. In essence, minimizing free-energy indirectly minimizes sensory surprisal, allowing organisms to adapt and survive in unpredictable and dynamic environments. Active Inference proposes to minimize free-energy by simultaneously 1) inferring the causes of sensory data and 2) acting in the world to align perceived sensations with desired sensations. The theory of Active Inference can be applied to both *continuous control* tasks and *discrete decision-making*, as explained next.

2.1.1. Continuous Active Inference

Active Inference has been applied for low-level control in robotics in several past works [109, 107, 98, 5]. Example applications are reaching tasks for which Active Inference computes low-level joint torques to reach a goal, see Figure 2.1. We now report the main ideas and derivations behind continuous Active Inference based on the free-energy principle, but we refer a reader to previous publications for more detailed explanations [109, 21].
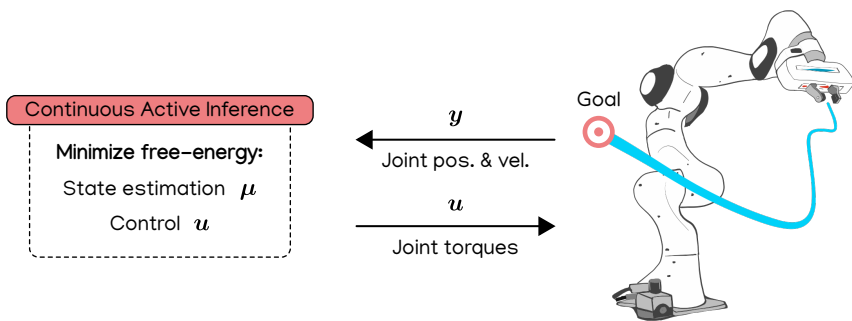
Figure 2.1: Illustrative idea of continuous Active Inference for low-level control. Given sensory information $\boldsymbol{y}$ about joint positions and velocities, continuous Active Inference estimates the current state $\boldsymbol{\mu}$ and computes control inputs $\boldsymbol{u}$ as joint torques by minimizing free-energy.

Free-energy principle

The free-energy principle is formulated in terms of Bayesian inference [83]. In fact, body perception for state estimation is framed using the Bayes rule:

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})} \tag{2.1}$$

where $p(\boldsymbol{x}|\boldsymbol{y})$ is the probability of being in the n-dimensional state $\boldsymbol{x}$ given the current m-dimensional sensory input $\boldsymbol{y}$. Instead of exactly inferring the posterior, which often involves intractable integrals, an auxiliary probability distribution $Q(\boldsymbol{x})$ is introduced. By minimizing the Kullback-Leibler divergence (D_{KL}) between the true posterior $p(\boldsymbol{x}|\boldsymbol{y})$ and $Q(\boldsymbol{x})$, the most probable state given a sensory input is inferred [21]. D_{KL} is defined as:

$$D_{KL}(Q(\boldsymbol{x})||p(\boldsymbol{x}|\boldsymbol{y})) = \int Q(\boldsymbol{x}) \ln \frac{Q(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y})} d\boldsymbol{x} = \mathcal{F} + \ln p(\boldsymbol{y}) \geq 0. \tag{2.2}$$

In the equation above, the scalar $\mathcal{F}$ is the so-called *free-energy*. By minimizing $\mathcal{F}$, D_{KL} is also minimized, and the $Q(\boldsymbol{x})$ approaches the true posterior. According to the Laplace approximation, the controller only parametrizes the sufficient statistics (e.g. mean and variance) of $Q(\boldsymbol{x})$ [49]. Furthermore, $Q(\boldsymbol{x})$ is assumed Gaussian and sharply peaked at its mean value $\boldsymbol{\mu}$. This approximation allows to simplify the expression for $\mathcal{F}$ which results:

$$\mathcal{F} \approx -\ln p(\boldsymbol{\mu}, \boldsymbol{y}). \tag{2.3}$$

The mean $\boldsymbol{\mu}$ is the internal belief about the true states $\boldsymbol{x}$. Minimizing $\mathcal{F}$, the controller is continuously adapting the internal belief $\boldsymbol{\mu}$ about the states $\boldsymbol{x}$ based on the current sensory input $\boldsymbol{y}$. Equation (2.3) is still general and must be further specified to numerically evaluate $\mathcal{F}$. To do so, the joint probability $p(\boldsymbol{\mu}, \boldsymbol{y})$ has to be defined. This is done by introducing two generative models, one to predict the sensory data $\boldsymbol{y}$, according to the current belief $\boldsymbol{\mu}$, and another to describe the dynamics of the evolution of the belief $\boldsymbol{\mu}$.

Generative model of the sensory data   The sensory data is modeled using the following expression [21]:

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{\mu}) + \boldsymbol{z} \tag{2.4}$$

where $\boldsymbol{g}(\boldsymbol{\mu})$ represents the non-linear mapping between sensory data and states of the environment, and $\boldsymbol{z}$ is Gaussian noise $\boldsymbol{z} \sim (\boldsymbol{0}, \Sigma_y)$. The covariance matrix Σ_y also represents the controller's confidence about each sensory input.

Generative model of the state dynamics   In the presence of time-varying states $\boldsymbol{x}$, the controller has to encode a dynamic generative model of the evolution $\boldsymbol{\mu}'$ of the belief $\boldsymbol{\mu}$. This generative model is defined as [21]:

$$\frac{d\boldsymbol{\mu}}{dt} = \boldsymbol{\mu}' = \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{w} \tag{2.5}$$

where $\boldsymbol{f}$ is a generative function dependent on the belief about the states $\boldsymbol{\mu}$ and $\boldsymbol{w}$ is Gaussian noise $\boldsymbol{w} \sim (\boldsymbol{0}, \Sigma_\mu)$.

2

Generalised motions To describe the dynamics of the states, or better the belief about these dynamics, we have to introduce the concept of generalised motions [52]. Generalised motions are used to represent the states of a dynamical system, using increasingly higher-order derivatives of the states of the system itself. They apply to sensory inputs as well, meaning that the generalised motions of a position measurement, for example, correspond to its higher-order temporal derivatives (velocity, acceleration, and so on). Using generalised motions allows a more accurate description of the system's states. More precisely, the generalised motions $\tilde{\boldsymbol{\mu}}$ of the belief under local linearity assumptions [49] are, up to the second order:

$$
\begin{aligned}
\boldsymbol{\mu}' &= \boldsymbol{\mu}^{(1)} = \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{w}, \\
\boldsymbol{\mu}'' &= \boldsymbol{\mu}^{(2)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}' + \boldsymbol{w}'.
\end{aligned}
\tag{2.6}
$$

In general, we indicate the generalised motions of the states up to order n_d[1] as $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}, \ \boldsymbol{\mu}', \ \boldsymbol{\mu}'', \ \boldsymbol{\mu}''', \ ..., \ \boldsymbol{\mu}^{(n_d)}]$. Similarly, the generalised motions of the sensory input are:

$$
\begin{aligned}
\boldsymbol{y} &= \boldsymbol{y}^{(0)} = \boldsymbol{g}(\boldsymbol{\mu}) + \boldsymbol{z}, \\
\boldsymbol{y}' &= \boldsymbol{y}^{(1)} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}' + \boldsymbol{z}'.
\end{aligned}
\tag{2.7}
$$

We indicate the generalised motions of the sensory input up to order n_d as $\tilde{\boldsymbol{y}} = [\boldsymbol{y}, \ \boldsymbol{y}', \ \boldsymbol{y}'', \ \boldsymbol{y}''', \ ..., \ \boldsymbol{y}^{(n_d)}]$.

General free-energy expression With the extra theoretical knowledge about the generalised motions, we can define an expression for the free-energy for a multivariate case in a dynamically changing environment:

$$
\mathcal{F} = -\ln p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{y}}).
\tag{2.8}
$$

The joint probability $p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{y}})$ has to be specified. Note that the noise at each dynamical order is considered uncorrelated [21]. Then, according to the generalised sensory input, the sensory data in a particular order relates only to the states in the same dynamical order. Similarly, for the state dynamics, the state at a certain dynamical order is related only to those which are one order below. Then, using the chain rule, it results:

$$
p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{y}}) = \prod_{i=0}^{n_d-1} p(\boldsymbol{y}^{(i)}|\boldsymbol{\mu}^{(i)}) p(\boldsymbol{\mu}^{(i+1)}|\boldsymbol{\mu}^{(i)}).
\tag{2.9}
$$

Using the Laplace assumption, and thus considering Gaussian distributed probability densities, we can write:

$$
\begin{aligned}
p(\boldsymbol{\mu}^{(i+1)}|\boldsymbol{\mu}^{(i)}) &= \frac{1}{|\Sigma_{\mu^{(i)}}|\sqrt[n]{2\pi}} \exp\left\{-\frac{1}{2}\boldsymbol{\varepsilon}_\mu^{(i)\top}\Sigma_{\mu^{(i)}}^{-1}\boldsymbol{\varepsilon}_\mu^{(i)}\right\} \\
p(\boldsymbol{y}^{(i)}|\boldsymbol{\mu}^{(i)}) &= \frac{1}{|\Sigma_{y^{(i)}}|\sqrt[n]{2\pi}} exp\left\{-\frac{1}{2}\boldsymbol{\varepsilon}_y^{(i)\top}\Sigma_{y^{(i)}}^{-1}\boldsymbol{\varepsilon}_y^{(i)}\right\}
\end{aligned}
\tag{2.10}
$$

---

[1]Generalised motions can extend up to infinite order but high orders are dominated by noise, thus we can limit the chosen order to n_d [44].

where $\varepsilon_y^{(i)} = (\boldsymbol{y}^{(i)} - \boldsymbol{g}^{(i)}(\boldsymbol{\mu}))$ and $\varepsilon_\mu^{(i)} = (\boldsymbol{\mu}^{(i+1)} - \boldsymbol{f}^{(i)}(\boldsymbol{\mu}))$ are respectively the sensory and state model prediction errors. Furthermore, it holds:

$$\boldsymbol{g}^{(i)} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)}, \quad \boldsymbol{f}^{(i)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)}, \quad \boldsymbol{g}^{(0)} = \boldsymbol{g}, \quad \boldsymbol{f}^{(0)} = \boldsymbol{f}. \tag{2.11}$$

Substituting (2.9) in (2.8) leads to:

$$\mathcal{F} = -\sum_{i=0}^{n_d-1} \left[\ln p(\boldsymbol{y}^{(i)}|\boldsymbol{\mu}^{(i)}) + \ln p(\boldsymbol{\mu}^{(i+1)}|\boldsymbol{\mu}^{(i)}) \right]. \tag{2.12}$$

Finally, according to (2.10), $\mathcal{F}$ can be expressed up to a constant as a weighted sum of squared prediction errors:

$$\mathcal{F} = \frac{1}{2} \sum_{i=0}^{n_d-1} \left[\varepsilon_y^{(i)\top} \Sigma_{y^{(i)}}^{-1} \varepsilon_y^{(i)} + \varepsilon_\mu^{(i)\top} \Sigma_{\mu^{(i)}}^{-1} \varepsilon_\mu^{(i)} \right] + K. \tag{2.13}$$

where n_d is the number of generalized motions chosen and K is a constant term resulting from the substitution. The minimization of this expression can be done by refining the internal belief, thus performing state estimation, but also computing the control actions to fulfill the prior expectations and achieve a desired motion. The constant term K is neglected in the sequel since it plays no role in the minimization problem. Next, we describe the core idea of Active Inference: minimizing $\mathcal{F}$, using gradient descent [45, 53].

Active Inference

Belief update for state estimation The belief update law for state estimation is determined from the gradient of the free-energy, with respect to each generalized motion [21, 52]:

$$\dot{\tilde{\boldsymbol{\mu}}} = \frac{d}{dt} \tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{\mu}}}. \tag{2.14}$$

The learning rate κ_μ, can be seen from a control perspective as a tuning parameter for the state update.

Control actions In the free-energy principle, the control actions play a fundamental role in the minimization process. In fact, the control input $\boldsymbol{u}$ allows to steer the system to a desired state while minimizing the prediction errors in $\mathcal{F}$. This is done using gradient descent. Since the free-energy is not a function of the control actions directly, but the actions $\boldsymbol{u}$ can influence $\mathcal{F}$ by modifying the sensory input, we can write [21]:

$$\frac{\partial \mathcal{F}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{y}}(\boldsymbol{u}))}{\partial \boldsymbol{u}} = \frac{\partial \tilde{\boldsymbol{y}}(\boldsymbol{u})}{\partial \boldsymbol{u}} \frac{\partial \mathcal{F}(\tilde{\boldsymbol{\mu}}, \boldsymbol{y}(\boldsymbol{u}))}{\partial \tilde{\boldsymbol{y}}(\boldsymbol{u})}. \tag{2.15}$$

Dropping the dependencies for a more compact notation, the dynamics of the control actions can be written as:

$$\dot{\boldsymbol{u}} = -\kappa_a \frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{u}} \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{y}}} \tag{2.16}$$

where κ_a is the tuning parameter to be chosen. We proposed to approximate the partial derivatives of the sensory input with respect to the control input with an identity mapping [109]. This led to a scheme that does not require precise dynamical information about a controlled system, but it presents some limitations that are detailed and addressed in Chapter 4.

2.1.2. Discrete Active Inference

In previous sections, we considered Active Inference in the context of continuous-time systems, where it has been cast as a gradient descent on free-energy. In this section, we briefly introduce how Active Inference can be used to explicitly model future discrete states and observations, thereby enabling *high-level planning*, see Figure 2.1 for an overview of the settings. Discrete action planning is crucial for many tasks and environments where actions have delayed consequences; thus, simply doing what is best in the current moment is not necessarily what is best in the long run. This section is kept at a higher level than the previous one since the full derivations of discrete Active Inference are treated in detail in Chapter 5.
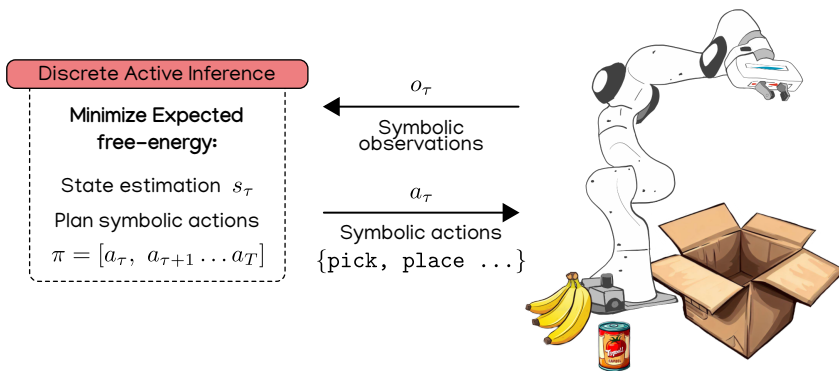
Figure 2.2: Illustrative idea of discrete Active Inference for high-level decision making. An example application could be to put all the products in the box given some `pick` and `place` skills. Given symbolic observations o_τ about the environment, discrete Active Inference estimates the current symbolic state s_τ and computes a plan π as a sequence of atomic actions that minimize expected free-energy over a horizon T. Then the first atomic action of the sequence a_τ is applied.

Models in discrete Active Inference

Modeling states and observations using generalized coordinates as described before does allow the system to incorporate knowledge of the future since generalized coordinates are a Taylor expansion in time around some given time point. In practice, to use generalized coordinates, it is necessary to truncate them at some small order, meaning they can only model smooth and local changes over time. Therefore, for reasons of mathematical tractability and computational efficiency, *discrete Active Inference* is introduced. It is assumed that the relationship between states, observations, and time are described by a Partially Observable Markov Decision

Process (POMDP) [69]. Intuitively, a POMDP assumes that there are discrete-time sequences of observations $\bar{o} = o_{1:T}$ and hidden states $\bar{s} = s_{1:T}$ up to some time horizon T. Now, states and observations do not represent continuous joint values as for the low-level control case. Instead, they represent discrete variables like holding an object or being in a specific room. This opens up the possibility of using Active Inference for high-level planning.

To do so, we need to define a model and its associated assumptions more concretely. At some given instant point τ, observations o_τ depend only on the hidden state s_τ. Similarly, it is assumed that the hidden states at the current time depend only on the hidden states at the previous time-step $s_{\tau-1}$ (Markov assumption) and the action at the previous time-step $a_{\tau-1}$. Thus, analogously to the continuous-time approach described in the previous sections, only current observation and previous state are needed to infer states and actions, instead of the entire history of states and observations. Mathematically, we consider a Markov process as the generative model P of the form:

$$P(\bar{o}, \bar{s}, \pi) = P(\pi) \prod_{\tau=1}^{T} P(s_\tau | s_{\tau-1}, \pi) P(o_\tau | s_\tau). \tag{2.17}$$

where $\pi = [a_\tau,\ a_{\tau+1} \ldots a_T]$ is a plan.

Expected free-energy

The model just introduced is used to derive a new concept to be able to plan for the future, namely *expected free-energy* [50, 104, 91]. The expected free-energy quantifies the average free-energy of a plan-conditioned trajectory of states and observations, rather than the instantaneous free-energy. Like in the continuous counterpart, discrete Active Inference agents are then mandated to minimize this quantity through both perception and action. An important terminological note concerns the word "*plan*" and its relation to the word policy. A plan is a sequence of actions $\pi = [a_\tau, a_{\tau+1} \ldots a_T]$. This differs from the policy used in reinforcement learning which refers to a parameterized action distribution conditioned on the current state.

The expected free-energy provides a measure quantifying the notion of the free-energy expected over future trajectories. In effect, it represents the average free-energy of a trajectory, taking into account the fact that because future observations are unknown they must be considered as random variables to be inferred, given a specific plan π. In the continuous-time formulation, goals are encoded as set points which, when compared against the current state estimates, form a prediction error that is minimized by action. In the POMDP formalism of discrete Active Inference, the goals or 'preferences' of the agent are encoded into the generative model to form a *biased generative model*.

Finding the optimal plan

One advantage of the expected free-energy is that it allows the derivation of an expression for the optimal plan over a trajectory in terms of the sum of the expected

free-energy for each time step. Discrete Active Inference addresses action and perception by assuming that actions fulfill predictions based on inferred states of the world from observations.

The generative model includes beliefs about future states and action plans, where plans leading to preferred observations are more probable. Perception and action are achieved by optimizing two complementary objective functions: the variational free-energy F and the expected free-energy G. These optimization objectives are derived from the generative model, as explained in detail in Chapter 5.

F quantifies the fit between the generative model and past/current sensory observations, while G evaluates potential future courses of action based on prior preferences and predicted observations. By optimizing these quantities, discrete Active Inference effectively balances action and perception, guiding the robot's behavior to achieve its goals while accounting for uncertainty and prediction accuracy.

Discrete Active Inference has been extensively studied for goal-directed and infor-mation-seeking behaviors [50, 47]. The resulting behaviors of discrete Active Inference agents have been related to human notions of curiosity and intrinsic motivation that produce exploratory behaviors. Artificial curiosity [119], intrinsic motivation [102] and goal-directed exploration [120] are crucial for learning and planning. We will explore the potential of Active Inference in Chapters 5 and 7 for reactive action planning and task and motion planning respectively.

2.2. Model Predictive Path Integral Control

This section presents the necessary background theory on Model Predictive Path Integral Control to understand the contributions of this thesis. MPPI was introduced in previous work [140, 141, 142, 143] as a solution for optimizing non-convex, nonlinear dynamics, for possibly high-dimensional systems. MPPI was initially based on a stochastic optimal control framework [141] and was addressed through path integral control theory [73]. Yet, this approach remained limited to control-affine systems. To broaden its applicability to a broader range of stochastic systems, the concept of information-theoretic MPPI [142, 143] emerged, eliminating the necessity for assuming control-affine systems. The background theory presented here is based on this formulation; for additional details, we refer the reader to the original paper [142]. A general idea of MPPI is visualized in Figure 2.3.

Preliminaries

Let's consider a stochastic dynamical system of the form

$$
\begin{aligned}
x_{t+1} &= f(x_t, v_t) \\
v_t &\sim \mathcal{N}(u_t, \Sigma),
\end{aligned}
\tag{2.18}
$$

where f is the nonlinear state transition function of a system, while $x_t \in \mathbb{R}^n$ and $v_t \in \mathbb{R}^m$ are the state and input at time t. We assumed that the actual input applied to the system v_t is affected by white noise and that we only have control over its mean u_t. Additionally, we assume a finite time horizon T such that $t \in \{0, 1, \ldots, T-1\}$.
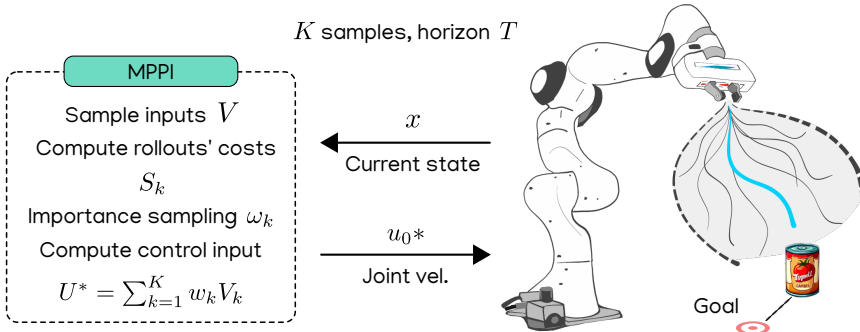
Figure 2.3: Illustrative idea of Model Predictive Path Integral Control, for instance, to push the can to a goal. MPPI takes the current state of the system x and computes a control input u_0^* to minimize a cost S_k. To do so, a number of control input sequences V are sampled and forward simulated up to a horizon T given a dynamic model. The resulting trajectory rollouts are then evaluated against the given cost function C and weighted through an importance sampling scheme. The result is one approximation of the optimal control input sequence. The first input of the sequence is then applied to the system and the process is repeated in a receding horizon fashion.

Thus, we define a sequence of inputs over the horizon as:

$$V = (v_0, v_1, \ldots, v_{T-1}) \in \mathbb{R}^{m \times T}$$
$$U = (u_0, u_1, \ldots, u_{T-1}) \in \mathbb{R}^{m \times T}. \tag{2.19}$$

We can then define the probability density function[2] for V as:

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

Given an input sequence V, the state trajectory can be obtained by recursively applying the state-transition function f from some initial conditions x_0. Thus, there exists a mapping from inputs V to trajectories $G_{x_0} : \Omega_V \to \Omega_\tau$ where $\Omega_\tau \subset \mathbb{R}^n \times \{0, \ldots, T-1\}$ represents the space of all possible trajectories. 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 written as:

$$U^* = \min_{U \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\phi(x_T) + \sum_{t=0}^{T-1} \mathcal{L}(x_t, u_t) \right]. \tag{2.21}$$

Assuming that the running cost can be split into a state-dependent running cost, and a control cost, we define:

$$\mathcal{L}(x_t, u_t) = c(x_t) + \frac{\beta}{2} \left(u_t^\top \Sigma^{-1} u_t + \zeta_t^\top u_t \right). \tag{2.22}$$

---

[2]For these derivations we will denote probability density functions with lowercase letters. The corresponding probability distributions are denoted by the same letter in uppercase *blackboard bold*. For example, the density $q(V|U, \Sigma)$ corresponds to the distribution $\mathbb{Q}_{U,\Sigma}$.

Now, considering only the part of the cost that depends on the state we define the following cost

$$C(x_0, x_1, \ldots, x_T) = \phi_{x_T} + \sum_{t=0}^{T-1} c(x_t). \tag{2.23}$$

Finally, given initial conditions x_0 and input sequence V, we can define a cost function $S(V) : \Omega_V \to \mathcal{R}^+$ as

$$S(x_0, V) = C \circ G_{x_0}. \tag{2.24}$$

For notation convenience, in the remainder of these derivations, we drop the initial conditions x_0 from the cost function definition unless explicitly required for clarity.

Information-theoretic framework

In Equation (2.21) we defined the form of the optimal control problem. The goal now is to determine an expression for an optimal control sequence. To achieve this, we will initially demonstrate the presence of a lower bound to the optimal control problem, which coincides with the free-energy of the system. Subsequently, we illustrate how this lower bound can be attained through a strategic selection of the optimal control distribution. This results in an information-theoretic framework for sampling-based model predictive control that will be used in Chapters 6 and 7.

Consider the following base distribution $\mathbb{P}$ with probability density function

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2} v_t^\top \Sigma^{-1} v_t\right). \tag{2.25}$$

Note that this is defined as the uncontrolled system dynamics, that is $\mathbb{P} = \mathbb{Q}_{0,\Sigma}$. Next, the free-energy[3] of the system from information theory is defined as [142]:

$$\mathcal{F}(S, \mathbb{P}, x_0, \beta) = -\beta \log \left(\mathbb{E}_{\mathbb{P}} \left[\exp \left(-\frac{1}{\beta} S(V) \right) \right] \right), \tag{2.26}$$

where $\beta \in \mathbb{R}^+$ is called inverse temperature. At this point, we can rewrite the expression of the free-energy above in terms of the controlled distribution $\mathbb{Q}_{U,\Sigma}$ by multiplying by $1 = \frac{q(V|U,\Sigma)}{q(V|U,\Sigma)}$, and then applying Jensen's inequality:

---

[3]Note that, in contrast to the free-energy used in Active Inference, here the free-energy refers to a mathematical quantity of a controlled system that takes the same form of Helmholtz free-energy. In the information-theoretic MPPI framework, this quantity is the starting point to define a lower bound on the optimal control problem. On the other hand, in Active Inference, the free-energy expression results from minimizing the surprisal of sensory inputs in order to survive. Although they are both called free-energy because of their similarities with thermodynamics, they are different and rely on different assumptions. It is not in the scope of this chapter, nor in this thesis, to discuss in detail the differences and similarities between the two formulations. The interested reader can find more details on the connection between path integral and information theory in previous publications [132, 131].

$$
\begin{aligned}
\mathcal{F}(S, \mathbb{P}, x_0, \beta) &= -\beta \log \left(\mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\exp \left(-\frac{1}{\beta} S(V) \right) \frac{p(V)}{q(V|U,\Sigma)} \right] \right) \\
&\leq -\beta \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\log \left(\exp \left(-\frac{1}{\beta} S(V) \right) \frac{p(V)}{q(V|U,\Sigma)} \right) \right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} [S(V)] + \beta \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\log \left(\frac{q(V|U,\Sigma)}{p(V)} \right) \right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} [S(V)] + \beta D_{KL}(\mathbb{Q}_{U,\Sigma} || \mathbb{P}).
\end{aligned}
\tag{2.27}
$$

Thus, we achieved the following relationship:

$$
\mathcal{F}(S, \mathbb{P}, x_0, \beta) \leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} [S(V)] + \beta D_{KL}(\mathbb{Q}_{U,\Sigma} || \mathbb{P}).
\tag{2.28}
$$

The right-hand side of Equation (2.28) contains the state cost for an optimal control problem and the KL-divergence between the base and controlled distributions. It can be seen how the free-energy serves as a lower bound on the sum of these two terms. Consider now the KL-divergence term. This can be simplified by considering the specific form of the probability density functions p and q from Equations (2.20) and (2.25). It holds that $D_{KL}(\mathbb{Q}_{U,\Sigma} || \mathbb{P})$ is equal to

$$
\begin{aligned}
&\mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\log \left(\prod_{t=0}^{T-1} \exp \left(-\frac{1}{2}(v_t - u_t)^\top \Sigma^{-1}(v_t - u_t) + \frac{1}{2} v_t^\top \Sigma^{-1} v_t \right) \right) \right] \\
=& \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\sum_{t=0}^{T-1} \left(-\frac{1}{2} u_t^\top \Sigma^{-1} u_t + \frac{1}{2} u_t^\top \Sigma^{-1} v_t + \frac{1}{2} v_t^\top \Sigma^{-1} u_t \right) \right].
\end{aligned}
\tag{2.29}
$$

Substituting into Equation (2.28), and recalling that Σ is symmetric and positive semi-definite, we can write:

$$
\begin{aligned}
\mathcal{F}(S, \mathbb{P}, x_0, \beta) &\leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} [S(V)] + \beta \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t + v_t^\top \Sigma^{-1} u_t \right) \right] \\
&= \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[S(V) + \sum_{t=0}^{T-1} \frac{\beta}{2} \left(u_t^\top \Sigma^{-1} u_t + 2 v_t^\top \Sigma^{-1} u_t \right) \right].
\end{aligned}
\tag{2.30}
$$

Now, by expanding $S(V)$, and by naming $\zeta_t^\top = 2 v_t^\top \Sigma^{-1}$ the following inequality holds:

$$
\mathcal{F}(S, \mathbb{P}, x_0, \beta) \leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}} \left[\phi(x_T) + \sum_{t=0}^{T-1} \left(c(x_t) + \frac{\beta}{2} \left(u_t^\top \Sigma^{-1} u_t + \zeta_t^\top u_t \right) \right) \right]
\tag{2.31}
$$

Note that we just showed that the free-energy is a lower bound for the objective of our optimal control problem since the right-hand side is precisely as in Equation (2.21).

Form of the optimal distribution

We just saw how the system's free-energy acts as a lower bound for the cost of an optimal control problem. We now show how, by choosing an appropriate distribution for control, we can achieve this lower bound. Take the optimal control density function $\mathbb{Q}^*$ as

$$q^*(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}S(V)\right) p(V), \tag{2.32}$$

where

$$\eta = \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\beta}S(V)\right)\right] = \int \exp\left(-\frac{1}{\beta}S(V)\right) p(V)dV. \tag{2.33}$$

We can now compute the KL-divergence between the base measure and controlled distribution as:

$$D_{KL}(\mathbb{Q}^*||\mathbb{P}) = \mathbb{E}_{Q^*}\left[\log \frac{\frac{1}{\eta}\exp\left(-\frac{1}{\beta}S(V)\right)p(V)}{p(V)}\right] = -\frac{1}{\beta}\mathbb{E}_{Q^*}\left[S(V)\right] - \log(\eta). \tag{2.34}$$

By substituting into Equation (2.28), and considering Equation (2.33), it results:

$$\mathcal{F} \leq \mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] - \mathbb{E}_{\mathbb{Q}^*}\left[S(V)\right] - \beta\log(\eta) = \underbrace{-\beta\log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\beta}S(V)\right)\right]\right)}_{\mathcal{F}(S,\mathbb{P},x_0,\beta)}. \tag{2.35}$$

The right-hand side is precisely the free-energy. Thus, inequality is reduced to equality, and the optimal distribution chosen achieves the lower bound. Thus, instead of minimizing Equation (2.21), we can push the controlled distribution $\mathbb{Q}_{U,\Sigma}$ close to $\mathbb{Q}^*$. If the former is close to the latter, then sampling from $\mathbb{Q}_{U,\Sigma}$ will result in low-cost trajectories.

Aligning the controlled distribution to the optimal one

We can align the controlled distribution to the optimal distribution by minimizing the following KL-divergence:

$$U^* = \min_{U \in \mathcal{U}}\left[D_{KL}\left(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}\right)\right]. \tag{2.36}$$

Because the optimal distribution is invariant to the particular control input, we can rewrite the problem above as:

$$\begin{aligned} U^* &= \min_{U \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{q^*(V)}{q(V|U,\Sigma)}\right)\right] \\ &= \min_{U \in \mathcal{U}}\left[\mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q^*(V)\right)\right] - \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right]\right] \\ &= \max_{U \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(q(V|U,\Sigma)\right)\right]. \end{aligned} \tag{2.37}$$

Now we can substitute the definition of $q(V|U, \Sigma)$ from Equation (2.20). This leads to:

$$
\begin{aligned}
U^* &= \max_{U \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}^*} \left[-\log \left(((2\pi)^m |\Sigma|)^{\frac{1}{2}} \right) - \frac{1}{2} \sum_{t=0}^{T-1} (v_t - u_t)^\top \Sigma^{-1} (v_t - u_t) \right] \\
&= \min_{U \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}^*} \left[\sum_{t=0}^{T-1} \left(\frac{1}{2} u_t^\top \Sigma^{-1} u_t - u_t^\top \Sigma^{-1} v_t \right) \right] \\
&= \min_{U \in \mathcal{U}} \left[\sum_{t=0}^{T-1} \frac{1}{2} u_t^\top \Sigma^{-1} u_t - \sum_{t=0}^{T-1} u_t^\top \Sigma^{-1} \mathbb{E}_{\mathbb{Q}^*} [v_t] \right]
\end{aligned}
\tag{2.38}
$$

In the unconstrained case, that is $\mathcal{U} = \mathbb{R}^m$, the objective above is concave and we can find the optimal solution by taking the gradient with respect to u_t and setting it to zero:

$$
\begin{aligned}
\Sigma^{-1} u_t^* - \Sigma^{-1} \mathbb{E}_{\mathbb{Q}^*}[v_t] &= 0, \text{ therefore} \\
u_t^* = \mathbb{E}_{\mathbb{Q}^*}[v_t] &= \int q^*(V) v_t dV, \\
\forall t &\in \{0, 1, \dots, T-1\}.
\end{aligned}
\tag{2.39}
$$

From the equation above, we can see that the optimal sequence is the expected value of control trajectories sampled from the optimal distribution. However, we cannot directly sample from the optimal distribution, and we need to introduce an approximation through importance sampling.

Importance sampling
Given an initial control input sequence U, we can write

$$
u_t^* = \int q^*(V) v_t dV = \int \underbrace{\frac{q^*(V)}{q(V|U, \Sigma)}}_{\omega(V)} q(V|U, \Sigma) v_t dV = \mathbb{E}_{\mathbb{Q}_{U, \Sigma}} \left[\omega(V) v_t \right],
\tag{2.40}
$$

where the term $\omega(V)$ is the importance sampling weight. Thanks to this term, we can compute the expectation with respect to $\mathbb{Q}^*$ by sampling trajectories from $\mathbb{Q}(V|U, \Sigma)$. We can further express the importance sampling weights as:

$$
\omega(V) = \frac{q^*(V)}{p(V)} \frac{p(V)}{q(V|U, \Sigma)} = \frac{1}{\eta} \exp \left(-\frac{1}{\beta} S(V) \right) \frac{p(V)}{q(V|U, \Sigma)}.
\tag{2.41}
$$

By substituting the terms $p(V)$ and $q(V|U, \Sigma)$ and performing a change of variable $v_t = u_t + \varepsilon_t$ we get:

$$
\begin{aligned}
\omega(V) &= \frac{1}{\eta} \exp \left(-\frac{1}{\beta} \left(S(V) + \frac{\beta}{2} \sum_{t=0}^{T-1} \left(u_t^\top \Sigma^{-1} u_t + 2\epsilon_t^\top \Sigma^{-1} u_t \right) \right) \right) \\
u_t^* &= \mathbb{E}_{\mathbb{Q}_{U, \Sigma}} \left[\omega(V) v_t \right].
\end{aligned}
\tag{2.42}
$$

The equation above describes the optimal information-theoretic control law for a given base distribution, even though in practice the expectation is evaluated through Monte-Carlo approximation. The presented theory can be used to define a sampling-based model predictive controller in a receding horizon fashion [142]. This is used in this thesis in Chapters 6 and 7 for robot control. Very briefly, the procedure initiates by observing the present state. Then, K trajectory samples are generated by evaluating randomized control sequences using forward dynamics. Then, the corresponding trajectory costs are computed. After that, the costs are used to compute the importance sampling weights and approximate the optimal control input sequence. The first input of the sequence is applied and the process is repeated. The next iteration of the control loop starts with the (shifted) previously computed input.

2.3. Fault-tolerant control

In the real world, technology is susceptible to faults and failures, making it crucial to design robust control systems capable of managing critical situations. A fault refers to a modification in a component's characteristics, leading to undesired changes in its performance or mode of operation. A failure denotes the system's or component's inability to fulfill its function, and once it occurs, it becomes irrecoverable. For a fault-tolerant controller, the primary goal is to prevent component faults from causing catastrophic failures. Developing fault-tolerant control schemes is of vital importance to bring robots outside controlled laboratories. The area of fault-tolerant control has become increasingly important in recent years, and several methods have been developed in different fields. Extensive bibliographical reviews and classifications of fault-tolerant methods can be found in the literature [146, 2]. In general, the design of a fault-tolerant controls scheme is comprised of two parts a) *fault diagnosis*: that is detection and isolation of a fault, and b) *fault recovery*: the adaptation of the controller parameters or structure to cope with the faulty situation. A general scheme is depicted in Figure 2.4.
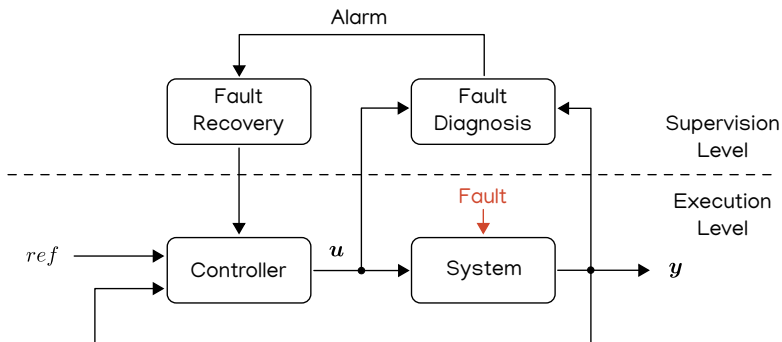
Figure 2.4: General structure of a fault-tolerant scheme [17] where a fault enters the controlled system.

Several approaches to fault-tolerant control are available. We consider here the

class of model-based techniques [29]. For fault detection, they rely on mathematical models to generate *residual* signals to be compared to a *threshold*. Fault recovery is then often performed by switching among different available fault-specific controllers [94]. The two main challenges to designing fault-tolerant schemes are the definition of residuals and thresholds, and the design of a fault-specific recovery strategy.

As can be seen in Figure 2.4, additional components must be integrated into the overall system to carry out fault detection, isolation, and controller adaptation for new situations. The diagnosis block specifically requires implementing a dynamic model that accounts for system behavior under faulty conditions. While some adaptive schemes can mitigate the need for external components, they are often limited in their applicability to specific fault types or gradual changes in process dynamics. We showed [107] how one can leverage the signals and parameters already present in an Active Inference controller to perform fault detection and recovery against sensory faults in robot manipulators. We proposed to use sensory prediction errors to build residuals for fault detection. Additionally, we used the sensory redundancy and precision matrices (or inverse covariance matrices) in the Active Inference controller for fault recovery. An overview of the main idea is depicted in Figure 2.5. The desired set-point $\boldsymbol{\mu}_d$ is provided to the Active Inference controller, which then
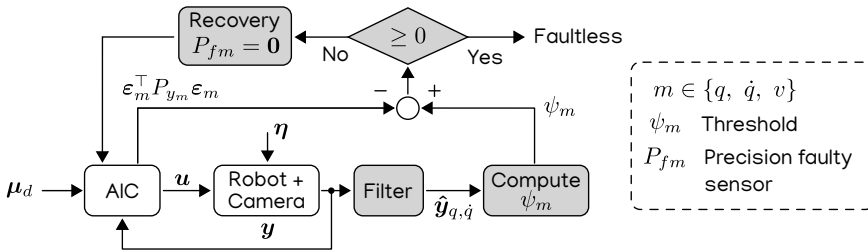
Figure 2.5: Fault-tolerant scheme using Active Inference [107]. Sensory prediction errors are compared to a computed threshold ψ_m from available data. When the threshold is exceeded, the precision (i.e. inverse covariance) $P_{fm} = \Sigma_{fm}^{-1}$ of a faulty sensor is set to zero to exclude it from the control loop.

computes a suitable control action $\boldsymbol{u}$. This is applied to a robot arm with sensory redundancy, namely encoders for the joint position and velocity and a camera for the end-effector. A threshold ψ_m is computed considering filtered joints' position and velocity. The sensory prediction errors are then compared to this threshold. If the threshold is exceeded, the associated precision P_{fm} of the faulty sensor is set to zero, to exclude the sensor from the control loop. For more details, we refer the interested reader to the full publication around this idea [107].

This Active Inference-based fault-tolerant scheme brings the advantages of simplified residual generation and fault recovery but is subject to false positives in case of rapidly changing set-points or collisions. These challenges are explained and addressed in Chapter 3.

3

Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference

*I*N *this chapter, we explore the use of Active Inference for low-level fault-tolerant control. Building upon our previous results on active Inference [107], we conduct an investigation into the limitations of past works and we propose a potential solution to address them. By doing so, we lay the foundations for the concept of unbiased Active Inference, which is further formalized in Chapter 4.*

3.1. Abstract

This work presents a novel fault-tolerant control scheme based on Active Inference. Specifically, a new formulation of Active Inference which, unlike previous solutions, provides unbiased state estimation and simplifies the definition of probabilistically robust thresholds for fault-tolerant control of robotic systems using the free-energy. The proposed solution makes use of the sensory prediction errors in the free-energy for the generation of residuals and thresholds for fault detection and isolation of sensory faults, and it does not require additional controllers for fault recovery. Results validating the benefits in a simulated 2-DOF manipulator are presented, and future directions to improve the current fault recovery approach are discussed.

3.2. Introduction

Fault-tolerant (FT) control is of vital importance for many applications such as robots working in production lines [75]. In order to guarantee high standards of reliability and security, the area of FT control of robot manipulators increasingly gathered importance in recent years [139]. Many approaches have been developed for faults in actuators and sensors [37, 68, 105, 136, 24]. Model-based fault-tolerant techniques are amongst the most advanced approaches to tackling the problem of fault detection and isolation (FDI) for dynamical systems [29]. These methods rely on monitoring system outputs using mathematical models to generate residual signals, which are compared to a threshold. Faults are detected if the threshold is exceeded, while the actions for fault recovery are usually performed through controller re-design or by switching to a different controller [94].

Regarding FT control for robot manipulators, Van and coauthors [136] perform fault diagnosis in image-based visual servoing. The residual signal is defined as the root mean squared estimation error (RMSE) of a Kalman filter, which predicts the values of a number of features in the camera image. The residual is compared against a user-defined static threshold and fault isolation is achieved via a decision table. The controller reconfiguration is not designed in the paper, and it would require an additional element on top of the FDI scheme. Visual information from a camera can also be used to compensate for a lack of observability in case of sensory faults. Capisani and coauthors [24] propose the use of three different second-order sliding mode observers for generating residuals using independent measurements from camera images. The detection thresholds were selected to minimize the probability of false or missed alarms on the basis of simulation and experimental data. However, the visual subsystem is assumed faultless.

FDI through static thresholds can be effective, but stochastic decision theory has been shown to outperform these methods. The actual distributions of residuals can be estimated such that a user can define thresholds based on their acceptable probability of false positives [40, 41, 114, 116]. The work of Blas and Blanke [18] showed fault-tolerant steering control of an agricultural vehicle for bailing, detecting sensory faults and artifacts in images through statistical learning. The sensor configuration for control was then decided at run-time.

Generally speaking, model-based fault-tolerant control schemes rely on addi-

tional supervision blocks that increase system complexity. FDI and fault recovery present two main challenges: a) The definition of robust yet sensitive pairs of residuals and thresholds; b) The design of specific additional supervisory systems to accommodate large faults.

The work presented in this paper extends the Active Inference controller (AIC) recently developed by the authors [109, 6]. Active inference relies on approximate Bayesian inference [45, 21] for prediction error minimization, and it allows state estimation and control using a single cost function. This scheme showed remarkable capabilities while dealing with missing sensory information or large unmodeled dynamics [77, 109] respectively. We initially presented a fault-tolerant control scheme with an AIC that showed great promise [107]. However, this presented a few fundamental limitations which will be thoroughly discussed in Section 3.3.2. Most importantly, FDI can be affected by false positives when the target state changes, and it relies on a static threshold which might be over-conservative. In this work, we: improve the state estimation of a standard AIC; develop an unbiased AIC (u-AIC); reduce the probability of false positives, and allow to easily define a probabilistically robust threshold for fault detection.

The paper is organized as follows: Section 3.3 presents the background on Active Inference for fault-tolerant control highlighting the limitations of past work. Section 3.4 presents a new formulation of the AIC with unbiased state estimation, while Section 3.5 explains how fault-tolerant control is achieved with this new formulation. Results for a simulated 2-DOF robot arm are presented in Section 3.6 while Section 3.7 reports a summary and future challenges.

3.3. Preliminaries

This section presents the background knowledge on Active Inference for robot control [109, 107] required to understand and justify the need for an unbiased AIC.

3.3.1. Active inference controller (standard AIC)

The AIC [109, 107] is defined for torque control in the joint space of a generic robot manipulator (Figure 3.1). The robot arm is equipped with encoders and velocity sensors with relative sensory readings $\boldsymbol{y}_q$, $\boldsymbol{y}_{\dot{q}}$. In addition, the end-effector Cartesian position $\boldsymbol{y}_v$ is made available through a camera. The system's output is represented by $\boldsymbol{y} = [\boldsymbol{y}_q, \ \boldsymbol{y}_{\dot{q}}, \ \boldsymbol{y}_v] \in \mathbb{R}^d$. The proprioceptive sensors and the camera are affected by zero mean Gaussian noise $\boldsymbol{\eta} = [\boldsymbol{\eta}_q, \ \boldsymbol{\eta}_{\dot{q}}, \ \boldsymbol{\eta}_v]$. The camera is affected by barrel distortion. The AIC can be used to control the robot arm to a desired configuration $\boldsymbol{\mu}_d$, providing the control input $\boldsymbol{u} \in \mathbb{R}^m$ as torques to the single joints. To this end, the control input is computed based on the so-called *free-energy* and the *generalised motions* $\tilde{\boldsymbol{\mu}}$ [107].

The free-energy principle relies on Bayesian inference [83] for state estimation. The goal is to find the posterior over states given observations, $p(\boldsymbol{x}|\boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}})$. This computation is in general intractable using Bayes' rule and in practice the true posterior distribution is approximated with a variational distribution $Q(\boldsymbol{x})$ [42]. The most probable state is computed by minimizing the Kullback-Leibler divergence
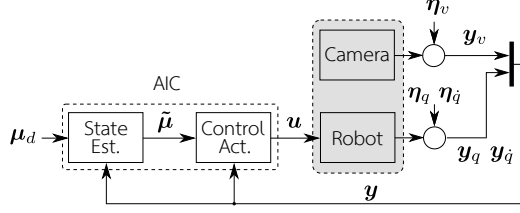
Figure 3.1: General AIC control scheme for joint space torque control.

3

(D_{KL}) between $p(\boldsymbol{x}|\boldsymbol{y})$ and $Q(\boldsymbol{x})$ [21]:

$$D_{KL}(Q(\boldsymbol{x})||p(\boldsymbol{x}|\boldsymbol{y})) = \int Q(\boldsymbol{x}) \ln \frac{Q(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y})} d\boldsymbol{x}$$
$$= \mathcal{F} + \ln p(\boldsymbol{y}) \tag{3.1}$$

where $\mathcal{F}$ represents the *free-energy*. By minimizing $\mathcal{F}$, $Q(\boldsymbol{x})$ approaches the true posterior $p(\boldsymbol{x}|\boldsymbol{y})$. The probabilistic model $p(\boldsymbol{x}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}})$ is factorized as:

$$p(\boldsymbol{x}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) = \underbrace{p(\boldsymbol{y}_v|\boldsymbol{x})p(\boldsymbol{y}_q|\boldsymbol{x})p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{3.2}$$

If $Q(\boldsymbol{x})$ is a Gaussian with mean $\boldsymbol{\mu}$ and the Laplace approximation is utilized [49], then $\mathcal{F}$ reduces to

$$\mathcal{F} = -\ln p(\boldsymbol{\mu}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) + K \,, \tag{3.3}$$

where K is a constant. Higher order derivatives of the state $\boldsymbol{x} \in \mathbb{R}^n$ are combined in the term $\tilde{\boldsymbol{\mu}}$ (i.e. $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}, \boldsymbol{\mu}', \boldsymbol{\mu}'']$). This is referred to as *generalized motions* [21, 48]. The number of generalised motions considered will affect the update laws for state estimation and control.

In order to characterize the free-energy and compute it, one has to define two generative models, one for the state dynamics, and one for the observations. The latter is modeled as [21] $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{\mu}) + \boldsymbol{z}$, where $\boldsymbol{g}(\boldsymbol{\mu}) = [\boldsymbol{g}_q, \ \boldsymbol{g}_{\dot{q}}, \ \boldsymbol{g}_v] : \mathbb{R}^d \mapsto \mathbb{R}^d$ represents the non-linear mapping between sensory data and states of the environment, and $\boldsymbol{z}$ is Gaussian noise $\boldsymbol{z} \sim (\boldsymbol{0}, \Sigma_y)$. The generative model of the state dynamics is defined as [21]:

$$\frac{d\boldsymbol{\mu}}{dt} = \boldsymbol{\mu}' = \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{w} \tag{3.4}$$

where $\boldsymbol{f}(\boldsymbol{\mu}) : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a generative function dependent on the belief about the states $\boldsymbol{\mu}$ and $\boldsymbol{w}$ is Gaussian noise $\boldsymbol{w} \sim (\boldsymbol{0}, \Sigma_\mu)$. As in previous work [6], we define $\boldsymbol{f}(\cdot)$ such that the robot will be steered to a desired joint configuration $\boldsymbol{\mu}_d$ following the dynamics of a first-order linear system with time constant τ.

$$\boldsymbol{f}(\boldsymbol{\mu}) = (\boldsymbol{\mu}_d - \boldsymbol{\mu})\tau^{-1} \tag{3.5}$$

From equation (3.5) we can see that the desired state is encoded in the prior. The time constant τ influences the generative model of the state dynamics $\boldsymbol{f}(\cdot)$.

As explained in past work [6], the AIC has two extremes depending on the value of τ^{-1}. If $\tau^{-1} \to 0$, the AIC only performs filtering and no control. On the other hand, if $\tau^{-1} \to \infty$ the AIC is equivalent to a PID controller [6, 12]. For any value in between, there is a compromise between estimation and control. The estimation and control are thus 'coupled' and state-estimation with the standard AIC results in a bias towards the desired target, see (3.7). If all distributions in equation (3.2) are assumed Gaussian, the expression for free-energy simplifies to (complete derivations can be found in past work [109]):

$$\mathcal{F} = \frac{1}{2} \sum_i \left(\boldsymbol{\varepsilon}_i^\top P_i \boldsymbol{\varepsilon}_i - \ln |P_i| \right) + K, \tag{3.6}$$

where $i \in \{y_q,\ y_{\dot{q}},\ y_v,\ \mu,\ \mu'\}$, and P_i defines a precision (or inverse covariance) matrix. Note that we set $\tau = 1$ as in past work [98, 109]. The terms $\boldsymbol{\varepsilon}_i$ with $i \in \{y_q,\ y_{\dot{q}},\ y_v\}$ are the *Sensory Prediction Errors* (SPE), representing the difference between observed sensory input and expected one. In general, the SPE for a sensor s are defined as $(\boldsymbol{y}_s - \boldsymbol{g}_s(\boldsymbol{\mu}))$. The model prediction errors are instead defined considering the desired state dynamics as $\boldsymbol{\varepsilon}_\mu = (\boldsymbol{\mu}' - \boldsymbol{f}(\boldsymbol{\mu}))$ and $\boldsymbol{\varepsilon}_{\mu'} = (\boldsymbol{\mu}'' - \partial \boldsymbol{f}(\boldsymbol{\mu})/\partial \boldsymbol{\mu} \boldsymbol{\mu}')$. In particular, for the robot arm used in this paper, $\boldsymbol{\varepsilon}_{y_q} = (\boldsymbol{y}_q - \boldsymbol{\mu})$, $\boldsymbol{\varepsilon}_{y_{\dot{q}}} = (\boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}')$, $\boldsymbol{\varepsilon}_v = (\boldsymbol{y}_v - \boldsymbol{g}_v(\boldsymbol{\mu}))$, and $\boldsymbol{\varepsilon}_\mu = (\boldsymbol{\mu}' + \boldsymbol{\mu} - \boldsymbol{\mu}_d)$, $\boldsymbol{\varepsilon}_{\mu'} = (\boldsymbol{\mu}'' + \boldsymbol{\mu}')$. For more details on the derivation of equation (3.6), refer to previous publications [109, 6, 21].

To achieve state estimation a gradient descent on the free-energy is used. The variable $\tilde{\boldsymbol{\mu}}$ is updated as:

$$\dot{\tilde{\boldsymbol{\mu}}} = D\tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{\mu}}}, \tag{3.7}$$

where κ_μ is the gradient descent step size, and D is a shifting operator with ones on the upper diagonal. This form of gradient descent is needed due to using generalized motions as mentioned earlier [45]. The control actions are also computed through gradient descent; however, the expression for $\mathcal{F}$ does not include any actions explicitly. The chain rule is then used, assuming an implicit dependency between actions and sensory input.

$$\dot{\boldsymbol{u}} = -\kappa_u \frac{\partial \mathcal{F}}{\partial \boldsymbol{u}} = -\kappa_u \frac{\partial \mathcal{F}}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}}, \tag{3.8}$$

where κ_u is the gradient descent's step size. The term $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}}$ is often approximated as linear [109, 98].

3.3.2. Limitations of fault-tolerant control with standard AIC

The AIC was previously used for fault-tolerant control [107]. In this subsection, we aim to highlight the main limitations of the AIC, which are addressed in Section 3.4 through the proposed unbiased Active Inference scheme. The main idea of AIC for fault-tolerant control was that the SPE in the free-energy could be used as residuals for fault detection, removing the need for more advanced residual generators. Besides, the precision matrices in the free-energy could be used for fault accommodation, since they represent the trust associated with sensory readings.

For a faulty sensor, the relative precision was decreased to zero to perform fault recovery. Specifically, the residuals are generated by considering the quadratic form of the SPE [107] $\boldsymbol{\varepsilon}_s^\top P_{y_s} \boldsymbol{\varepsilon}_s$, where $\boldsymbol{\varepsilon}_s = \boldsymbol{y}_s - \boldsymbol{g}_s(\boldsymbol{\mu})$ with $s \in \{q, \dot{q}, v\}$. A static fault detection threshold ψ_s was chosen as an upper bound on the quadratic terms.

The SPE depends on the belief $\boldsymbol{\mu}$, which is biased towards a given goal. This means that the SPE in standard AIC can increase for causes that are not necessarily related to a fault, i.e. the robot is stuck due to a collision or there is an abrupt change in the goal $\boldsymbol{\mu}_d$, for example, in a pick-and-place application. The residuals are thus dependent on the goal, which is problematic. This was not an issue in past work [107] because of the over-conservative threshold used for fault detection, but it becomes apparent when the SPEs are closely analyzed. Figure 3.2 reports the absolute value of the SPE for the joint of a robot arm controlled using an AIC, during a point-to-point motion with three via-points $\boldsymbol{\mu}_{d_1}$, $\boldsymbol{\mu}_{d_2}$, $\boldsymbol{\mu}_{d_3}$.
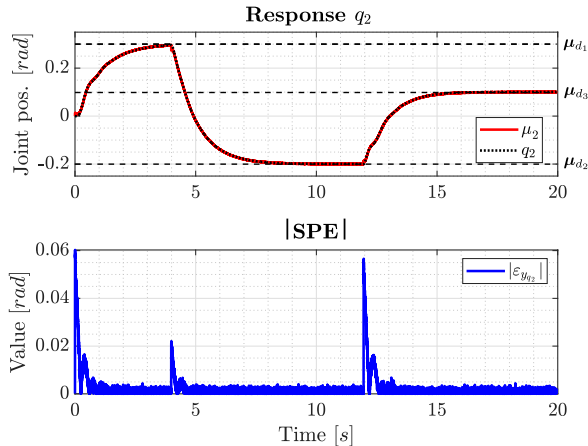
Figure 3.2: Effect of abrupt change in goal on the SPE in standard AIC. The SPE increases sharply without the presence of any sensory faults.

Using a tighter threshold to detect faults of small entities by using past work [107] would lead to many false positives. In this work, we tackle the problem of biased sensory prediction errors and we define an *unbiased AIC* where the SPEs are independent of the input, providing more accurate state estimation. With the u-AIC is also possible to use the sensory prediction error directly for fault detection using a probabilistically robust threshold instead of a static one as in the previous approach [107].

3.4. Unbiased Active Inference

To tackle the problems discussed in the previous section, the unbiased AIC is developed (u-AIC).

3.4.1. Derivation of the u-AIC

Consider a probabilistic model with explicit actions $p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}})$, where unlike the AIC, $\boldsymbol{x} = [\boldsymbol{q} \ \dot{\boldsymbol{q}}]^\top$. The distribution factorizes as:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \underbrace{p(\boldsymbol{y}_v|\boldsymbol{x})p(\boldsymbol{y}_q|\boldsymbol{x})p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{3.9}$$

Given the sensory data, we then aim to find the posterior over states and actions $p(\boldsymbol{x}, \boldsymbol{u}|\boldsymbol{y}_v, \boldsymbol{y}_q)$. We approximate the posterior with a variational distribution $Q(\boldsymbol{x}, \boldsymbol{u})$. We utilize the mean-field assumption ($Q(\boldsymbol{x}, \boldsymbol{u}) = Q(\boldsymbol{x})Q(\boldsymbol{u})$) and the Laplace approximation. The posterior over the state $\boldsymbol{x}$ is assumed Gaussian with mean $\boldsymbol{\mu}_x$. The posterior over actions $\boldsymbol{u}$ is also assumed Gaussian with mean $\boldsymbol{\mu}_u$. This results in the following expression for variational free-energy:

$$\mathcal{F} = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) + C \tag{3.10}$$

This expression can be factorized as in equation (3.9). Assuming Gaussian model $\mathcal{F}$ can be expanded to:

$$\begin{aligned}
\mathcal{F} = \frac{1}{2}(\boldsymbol{\varepsilon}_{y_q}^\top P_{y_q} \boldsymbol{\varepsilon}_{y_q} + \boldsymbol{\varepsilon}_{y_{\dot{q}}}^\top P_{y_{\dot{q}}} \boldsymbol{\varepsilon}_{y_{\dot{q}}} + \boldsymbol{\varepsilon}_{y_v}^\top P_{y_v} \boldsymbol{\varepsilon}_{y_v} \\
+ \boldsymbol{\varepsilon}_x^\top P_x \boldsymbol{\varepsilon}_x + \boldsymbol{\varepsilon}_u^\top P_u \boldsymbol{\varepsilon}_u - \ln|P_u P_{y_q} P_{y_{\dot{q}}} P_{y_v} P_x|) + C,
\end{aligned} \tag{3.11}$$

where $\boldsymbol{\varepsilon}_{y_q}$, $\boldsymbol{\varepsilon}_{y_{\dot{q}}}$, $\boldsymbol{\varepsilon}_{y_v}$ are the sensory prediction errors of position encoder, velocity encoder, and the visual sensor respectively. Furthermore, $\boldsymbol{\varepsilon}_x$ and $\boldsymbol{\varepsilon}_u$ are the prediction errors for the prior on the state and the control action respectively.

The prediction error on the state prior $\boldsymbol{\varepsilon}_x$ is computed considering a prediction of the state $\hat{\boldsymbol{x}}$ at the current time-step, which is a deterministic value. It follows that $\boldsymbol{\varepsilon}_x = (\boldsymbol{\mu}_x - \hat{\boldsymbol{x}})$. The prediction $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{q}} \ \hat{\dot{\boldsymbol{q}}}]^\top$ can be computed in the same fashion as the prediction step of, for instance, a Kalman filter or of a Luenberger observer. While in such cases the prediction computation would require the knowledge of an accurate dynamic model, in this paper we assume we do not have access to that. Instead, we will propagate forward in time the current state belief using the following simplified discrete time model:

$$\hat{\boldsymbol{x}}_{k+1} = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \boldsymbol{\mu}_{x,k} \tag{3.12}$$

where I represents a unitary matrix of suitable size. This form assumes that the velocities of the joints will remain roughly constant, and the position of each joint is thus computed as the discrete-time integral of the velocity, using a first-order Euler scheme. As mentioned, if one has access to a better dynamic model, that can be used instead.

Finally, the state prior now does not encode information about the target $\boldsymbol{\mu}_d$ (unlike in the standard AIC). The information about the target is encoded in the distribution $p(\boldsymbol{u}|\boldsymbol{x})$. We choose this distribution to be Gaussian as well with a mean of $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$, which is a function that steers the systems toward the target.

This then results in $\boldsymbol{\varepsilon}_u = (\boldsymbol{\mu}_u - f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d))$. The function $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$ can be *any* controller, for instance a P controller: $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)) = P(\boldsymbol{\mu}_d - \boldsymbol{\mu}_x)$. In this paper, we choose the function such that it converges to a PID controller; however, other choices can be made without loss of generality. For state and action updates, first-order Euler integration is used to obtain discrete time equations.

3.4.2. Definition of the observation model

The sensory prediction errors are as in the AIC, that is $\boldsymbol{\varepsilon}_q = (\boldsymbol{y}_q - \boldsymbol{\mu})$, $\boldsymbol{\varepsilon}_{\dot{q}} = (\boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}')$ and $\boldsymbol{\varepsilon}_v = (\boldsymbol{y}_v - \boldsymbol{g}_v(\boldsymbol{\mu}))$ but $\boldsymbol{\mu}$ is not biased anymore towards the target $\boldsymbol{\mu}_d$. The relation between $\boldsymbol{\mu}$ and $\boldsymbol{y}$ is expressed through the generative model of the sensory input $\boldsymbol{g} = [\boldsymbol{g}_q, \ \boldsymbol{g}_{\dot{q}}, \ \boldsymbol{g}_v]$. Position and velocity encoders directly measure the state. Thus $\boldsymbol{g}_q$ and $\boldsymbol{g}_{\dot{q}}$ are linear mappings between $\boldsymbol{\mu}$ and $\boldsymbol{y}$.

To define $\boldsymbol{g}_v(\boldsymbol{\mu})$, instead, we use a *Gaussian Process Regression* (GPR) as in past work [77]. This is particularly useful because we can model the noisy and distorted sensory input from the camera, and at the same time, we can compute a closed form for the derivative of the process with respect to the beliefs $\boldsymbol{\mu}$, required for the state update laws. The training data is composed of a set of observations of the (planar) camera output $[\bar{\boldsymbol{y}}_{v_x}, \ \bar{\boldsymbol{y}}_{v_z}]^\top$ in several robot configurations $\bar{\boldsymbol{y}}_q$. We use a squared exponential kernel k of the form:

$$k(\boldsymbol{y}_{q_i}, \boldsymbol{y}_{q_j}) \quad = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{y}_{q_i} - \boldsymbol{y}_{q_j})^\top \Theta(\boldsymbol{y}_{q_i} - \boldsymbol{y}_{q_j})\right)$$
$$+ \sigma_n^2 d_{ij} \tag{3.13}$$

where $\boldsymbol{y}_{q_i}, \ \boldsymbol{y}_{q_j} \in \bar{\boldsymbol{y}}_q$, and d_{ij} is the Kronecker delta function. Θ is a diagonal matrix of hyperparameters that are fit according to the training data to obtain accurate predictions. The hyperparameters are optimized in order to maximize the marginal likelihood [112]. The predictions are then [77]:

$$\boldsymbol{g}_v(\boldsymbol{y}_{q_*}) = \begin{bmatrix} k(\boldsymbol{y}_{q_*}, \bar{\boldsymbol{y}}_q)K^{-1}\bar{\boldsymbol{y}}_{v_x} \\ k(\boldsymbol{y}_{q_*}, \bar{\boldsymbol{y}}_q)K^{-1}\bar{\boldsymbol{y}}_{v_z} \end{bmatrix}$$
$$\boldsymbol{g}_v(\boldsymbol{y}_{q_*})' = \begin{bmatrix} -\Theta^{-1}(\boldsymbol{y}_{q_*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{q_*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_x] \\ -\Theta^{-1}(\boldsymbol{y}_{q_*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{q_*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_z] \end{bmatrix} \tag{3.14}$$

where $\cdot$ means element-wise multiplication, K is the covariance matrix, $\boldsymbol{\alpha}_x = K^{-1}\bar{\boldsymbol{y}}_{v_x}$ and $\boldsymbol{\alpha}_z = K^{-1}\bar{\boldsymbol{y}}_{v_z}$.

3.4.3. Estimation and control using the u-AIC

Similar to the AIC, performing both state estimation and control can be achieved using gradient descent on $\mathcal{F}$. This is simpler in the case of the u-AIC since there is no need to use the chain rule when computing the control actions (see eq. (3.8)) or to consider generalized motion (3.7):

$$\dot{\boldsymbol{\mu}}_u = -\kappa_u \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_u}, \quad \dot{\boldsymbol{\mu}}_x = -\kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_x}, \tag{3.15}$$

where κ_u and κ_μ are the gradient descent step sizes. The gradient on control can

be computed as:

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_u} = P_u(\boldsymbol{\mu}_u - f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)) \tag{3.16}$$

Setting this to zero results in the control action being equal to $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$. If this function is chosen similarly to a PID controller, then the control law of our system will converge towards that. This can be extended to richer control by modifying the probabilistic model in eq. (3.9). The only term depending on the control action now is $p(\boldsymbol{u}|\boldsymbol{x})$; however, a prior $p(\boldsymbol{u})$ can also be added. In that case, the generative model would be:

$$\frac{1}{\alpha} p(\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{x}) p(\boldsymbol{y}_v|\boldsymbol{x}) p(\boldsymbol{y}_q|\boldsymbol{x}) p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x}) p(\boldsymbol{x}) \tag{3.17}$$

where α is a normalization constant. This prior can then encode a feed-forward signal (open-loop control law). When computing the $\boldsymbol{u}$ that minimizes $\mathcal{F}$ in that case, it would be a combination of the feed-forward signal and $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$ depending on the value of Σ_u. This is employed by Baioumy and coauthors [5].

Since $\boldsymbol{\mu}_u$ converges to $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$, we can achieve unbiased state estimation. In fact, this would result in $\boldsymbol{\varepsilon}_u = 0$, and thus the current target would not affect the the beliefs about the states. In u-AIC, the belief about the states is only affected by the predicted values and the observations from various sensors, exactly as in a Kalman filter [88].

3.4.4. Key differences between AIC and u-AIC
The AIC considers the relationship between states and observations without explicitly modeling the control actions. The prior over state $p(\boldsymbol{x})$ in equation (3.2) thus encodes the target/goals state. In filters, such as the Kalman filter, the prior comes from a prediction step that relies on the previous state and action. In the case of the AIC, the beliefs update law is essentially a filter where the prediction step is biased towards the goals state i.e. the agent normally predicts to move linearly towards the target (see eq. (3.5)).

In the u-AIC, actions are explicitly modeled and the target state is encoded in the term $p(\boldsymbol{u}|\boldsymbol{x})$. This has two implications. First, the state is not biased towards the target since its prior $p(\boldsymbol{x})$ is not. A prior encoding of a prediction step will also improve the overall state estimation accuracy. Secondly, explicit actions allow us to directly perform gradient descent on $\mathcal{F}$ with respect to $\boldsymbol{u}$. This is not ppossible in the general AIC case and the chain rule had to be utilized (see eq. (3.8)).

Finally, the u-AIC has guarantees regarding the stability of the controller, while the AIC does not. This is out of the scope of this paper and will be discussed in future work.

3.5. Fault detection, isolation, and recovery
In this section, we describe how the SPE from the u-AIC can be used as residual signals, and how to perform FDI. Without loss of generality, for these derivations, we will consider only the proprioceptive sensors and we discretize the continuous dynamics with first-order Euler integration. According to the approximated system's

dynamics in (3.12), and expanding the second term in (3.15) where $\mathcal{F}$ is computed as in (3.11), the state estimation law can be rewritten as:

$$\boldsymbol{\mu}_{k+1} = \gamma(\boldsymbol{\mu}_k, \boldsymbol{u}_k) + \Lambda(\boldsymbol{g}(\boldsymbol{\mu}_k) - \boldsymbol{y}_k) \tag{3.18}$$

Equation (3.18) represents the dynamics of an estimator where stability results from the value that we obtain for diagonal matrix Λ. Anil Meera and Wisse [88] showed how the free-energy principle can be used to derive stable state observers, for a linear case with colored noise. It is important to note that γ and Λ are known expressions resulting from the partial derivatives of $\mathcal{F}$. The term $(\boldsymbol{y}_k - \boldsymbol{g}(\boldsymbol{\mu}_k))$ represents the sensory prediction errors. In the u-AIC, the states of the system are steered towards the desired goal following the dynamics imposed by the controller. Therefore, the resulting system's dynamics can be represented in general as:

$$\begin{cases} \boldsymbol{x}_{k+1} = \gamma(\boldsymbol{x}_k, \boldsymbol{u}_k) + \phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho}_k) \\ \boldsymbol{y}_k = \boldsymbol{x}_k + \boldsymbol{z}_k \end{cases} \tag{3.19}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\boldsymbol{u}_k \in \mathbb{R}^m$ are the state and input variables, while $\gamma(\boldsymbol{x}_k, \boldsymbol{u}_k) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ represents the dynamics of the system in healthy conditions. $\boldsymbol{y}_k \in \mathbb{R}^n$ is the measurement of the full state which is affected by the presence of measurement noise $\boldsymbol{z}_k \in \mathbb{R}^n$. The expressions obtained so far for the system dynamics will allow us to cast a model-based fault diagnosis approach easily in the current framework, even if the knowledge of an a priori model is not needed. The real system can be affected by faults which are represented by the fault function $\phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho}_k) : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \mapsto \mathbb{R}^n$. The unknown parameter $\boldsymbol{\rho}_k$ determines the fault amplitude and shall be such that $\phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{0}) = 0$.

3.5.1. Residual generation

We rely on two assumptions that can be found as well in past works [114, 115]:

Assumption 1. *No fault acts on the system before the fault time k_f, thus $\boldsymbol{\rho}_k = 0$ for $0 \leq k < k_f$. In addition, before and and after the occurrence of a fault, the variables $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ remain bounded, that is $\exists \, \mathcal{S} = \mathcal{S}^{\boldsymbol{x}} \times \mathcal{S}^{\boldsymbol{u}} \subset \mathbb{R}^n \times \mathbb{R}^m$ such that $(\boldsymbol{x}_k, \, \boldsymbol{u}_k) \in \mathcal{S} \, \forall k$.*

Assumption 2. *$\boldsymbol{z}_k$ is a random variable defined on some probability spaces $(\mathcal{Z}, \mathcal{B}, \mathbf{P}_{\mathcal{Z}})$, where $\mathcal{Z} \subseteq \mathbb{R}^n$. $\mathcal{B}(\cdot)$ indicates a Borel σ-algebra, and $\mathbf{P}_{\mathcal{Z}}$ is the probability measures defined over $\mathcal{Z}$. Furthermore, it is not correlated and is independent of $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ and $\boldsymbol{\rho}_k \, \forall k$.*

We define the residuals for fault detection as the sensory prediction errors $\boldsymbol{r}_k = \boldsymbol{y}_k - \boldsymbol{g}(\boldsymbol{\mu}_k)$. Thus, according to (3.19), (3.18), the residuals dynamics will be given by [114, 115]:

$$\boldsymbol{r}_{k+1} = \Lambda \boldsymbol{r}_k + \boldsymbol{\delta}_k + \phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho}_k) \tag{3.20}$$

$$\boldsymbol{\delta}_k = \gamma(\boldsymbol{y}_k - \boldsymbol{z}_k, \boldsymbol{u}_k) - \gamma(\boldsymbol{\mu}_k, \boldsymbol{u}_k) + \boldsymbol{z}_{k+1} \tag{3.21}$$

The stochastic process $\boldsymbol{\delta}_k$, is the random total uncertainty that affects the residuals generation. It follows that $\boldsymbol{\delta}_k$ is in the probability space $(\Delta_k, \mathscr{B}(\Delta_k), \mathbf{P}_{\boldsymbol{\delta}_k})$ where Δ_k is obtained by letting $\boldsymbol{z}_k$ and $\boldsymbol{z}_{k+1}$ vary over $\mathcal{Z}$. Apart from simple cases, it is not possible to obtain a closed form for this set. Thus numerical approximations are used instead. The residual $\boldsymbol{r}_{k+1}$ can be seen as a random variable in the same probability space of $\boldsymbol{\delta}_k$ [114].
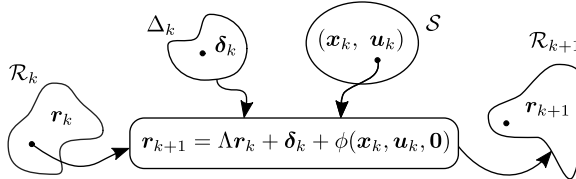
Figure 3.3: Healthy residual set at time $k+1$ as image obtained from the output of (3.20).

The residual set $\mathcal{R}_{k+1}$ at the next time step $k+1$ can be seen as the image obtained by computing the output of (3.20), where the total uncertainty $\boldsymbol{\delta}_k$ varies over its domain Δ_k (Figure 3.3). Note that the healthy residuals can be characterized by setting $\boldsymbol{\rho}_k = 0$.

Remark 1. *While the system dynamics γ and the fault function ϕ have been introduced for analysis purposes, they do not correspond to known models of healthy and faulty behaviors. In particular, γ is obtained via algebraic manipulations from the current free-energy framework's update equations. In particular, the residual r for implementing the detection and isolation logic described next can be directly computed from the current belief and measurement. Indeed, the proposed fault diagnosis approach based on the u-AIC is completely model-free.*

3.5.2. Threshold for fault detection

As proposed by Rostampour and coauthors [115], we consider a probabilistically robust detection logic of the form:

$$d_M(\boldsymbol{r}_{k+1}) \leq \frac{n}{\alpha} \triangleq \overline{d_M} \tag{3.22}$$

where $\overline{d_M}$ is the detection threshold, n is the dimension of the residuals, α is a tuning parameter, and $d_M(\cdot)$ indicates the Mahalanobis distance of the residuals. In general:

$$d_M(\boldsymbol{r}) = \sqrt{(\boldsymbol{r} - \bar{\boldsymbol{r}})^\top C_r^{-1}(\boldsymbol{r} - \bar{\boldsymbol{r}})} \tag{3.23}$$

where $\bar{\boldsymbol{r}} \triangleq \mathbb{E}[\boldsymbol{r}] \in \mathbb{R}^n$ and $C_r \triangleq Cov[\boldsymbol{r}] \in \mathbb{R}^{n \times n}$ are the expected value and covariance matrix of the residuals. According to the Multivariate Chebyshev Inequality [30]:

$$Pr[d_M(\boldsymbol{r}_k) \geq \frac{n}{\alpha}] \leq \alpha, \quad \forall \alpha \in [0, 1] \tag{3.24}$$

This means that, in healthy conditions, the probability of a false alarm, that is d_M exceeding the threshold $\bar{d}_M$, is upper bounded by α. The value α can be tuned to

achieve the desired probabilistic robustness of the threshold. The present detection logic is equivalent to checking if the residual at time step $k+1$ belongs to a time-varying ellipsoid with mean $\bar{\boldsymbol{r}}_{k+1}$ and covariance $C_{r_{k+1}}$. In the general nonlinear case, these moments can be approximated by their sampled counterparts obtained in healthy conditions.

3.5.3. Fault isolation

The detection policy in (3.22) results effective to label the system as healthy or faulty. However, it is not possible to use the same SPE generated through equation (3.11) for fault isolation. In fact, the free-energy is computed as a weighted sum of the squared prediction errors fusing different sensory sources. This means that when a fault occurs, its effects will propagate to all the SPE in equation (3.11). We define two additional free-energies and state estimation processes that rely on different sensory sources (i.e. either proprioceptive or visual).

$$
\begin{aligned}
\mathcal{F}_p = \frac{1}{2}(\boldsymbol{\varepsilon}_{\boldsymbol{y_q}}^\top P_{y_q}\boldsymbol{\varepsilon}_{\boldsymbol{y_q}} + \boldsymbol{\varepsilon}_{\boldsymbol{y_{\dot{q}}}}^\top P_{y_{\dot{q}}}\boldsymbol{\varepsilon}_{\boldsymbol{y_{\dot{q}}}} + \boldsymbol{\varepsilon}_{\boldsymbol{x}}^\top P_x\boldsymbol{\varepsilon}_{\boldsymbol{x}} \\
- \ln|P_{y_q}P_{y_{\dot{q}}}P_x|),
\end{aligned}
\tag{3.25}
$$

$$
\mathcal{F}_v = \frac{1}{2}(\boldsymbol{\varepsilon}_{\boldsymbol{y_v}}^\top P_{y_v}\boldsymbol{\varepsilon}_{\boldsymbol{y_v}} + \boldsymbol{\varepsilon}_{\boldsymbol{x}}^\top P_x\boldsymbol{\varepsilon}_{\boldsymbol{x}} - \ln|P_{y_v}P_x|)
\tag{3.26}
$$

By doing so, we can isolate encoder and camera faults since state estimation is now done using two independent measurement sources. $\mathcal{F}_p$ and $\mathcal{F}_v$ are only used for fault isolation and not control. The thresholds for the SPE from (3.25) and (3.26) are then computed as in Section 3.5.2.

3.5.4. Fault recovery

Fault recovery can be performed as in previous work [107]. Once a fault is detected and isolated, fault recovery is triggered. If a sensor is marked as faulty, its corresponding precision matrix is set to zero. The controller can thus exploit the sensory redundancy to a) have a better a posteriori approximation of the states, and b) compensate for missing or wrong sensory data. Once a fault is detected and isolated, the precision matrix of the faulty sensor P_{fs} is reduced to zero:

$$
P_{fs} = \boldsymbol{0}
\tag{3.27}
$$

Interestingly, the Active Inference framework also allows hyper-parameters (precision) learning [6, 19]. The bias in the standard AIC hindered hyper-parameter learning for fault recovery purposes, but learning meaningful precision matrices associated with sensory readings with u-AIC is now possible and will be investigated in future work.

3.6. Simulation study

In this section, we illustrate the fault detection, isolation, and recovery strategy on a MATLAB simulation using a 2-DOF robotic manipulator. The dynamical model

of the robot is defined as [124]:

$$u = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + D\dot{q} + G(q) \tag{3.28}$$

where $q = [q_1,\ q_2]^\top$, $u(t) = [u_1,\ u_2]^\top$, D is the friction coefficient matrix, M is the inertia matrix, C is the Coriolis matrix, and G models the effects of gravity. The dynamic model is only used to simulate the response to the commanded torques since the u-AIC provides a control law agnostic to the dynamical model. Thus, the link's masses, friction coefficients, and other dynamical parameters are not part of the controller. The noisy sensory readings have zero-mean Gaussian noise. The standard deviation of the noise for encoders and velocity sensors is set to $\sigma_q = \sigma_{\dot{q}} = 0.001$, while the one for the camera to $\sigma_v = 0.01$.

The simulation consists of controlling the robot arm for a double point-to-point motion from the starting configuration $q = [-\pi/2,\ 0]\ (rad)$ to $\mu_{d1} = [-0.2,\ 0.5]\ (rad)$, and then $\mu_{d2} = [-0.6,\ 0.2]\ (rad)$. A fault occurs during the trajectory from μ_{d1} to μ_{d2}, i.e. we set $k_f = 8s$. The fault can affect either the encoders or the camera. The encoder fault is such that the output related to the first joint freezes so $y_q(k) = [q_1(k_f),\ q_2(k)]^\top$ for $k \geq k_f$. The fault detection and recovery of such a fault, as well as the system's response, are reported below in Figure 3.4 and Figure 3.5:
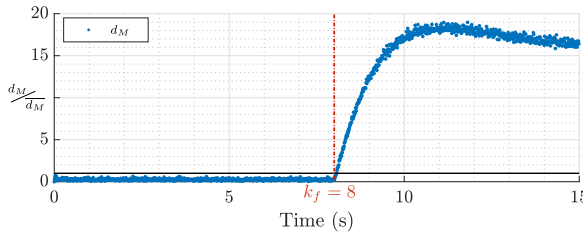
Figure 3.4: Normalised d_M in case of encoder fault. Detection occurs when the normalized value exceeds 1.

As can be seen from Figure 3.5, the system is not able to reach the set-point after the occurrence of the fault in case of no recovery. The robot arm reaches a different configuration to minimize the free-energy, which is built fusing the sensory information from the (faulty) encoders and the (healthy) camera. Recovery happens after $60\ (ms)$, and after that, the faulty reading is discarded by setting zero precision. The robot arm is then able to reach the final configuration.

The other situation that we consider in this work is a camera fault. This fault is supposed to be a misalignment, due for instance to a collision. This is simulated by injecting a bias i.e. $y_v(k) = y_v(k) + 0.04$ for $k \geq k_f$. Note that the entity of this fault is small and comparable with the camera noise, i.e. the zero mean Gaussian noise with $\sigma_v = 0.01\ (m)$. The fault detection is reported in Figure 3.6. The system's response with and without recovery is similar to Figure 3.5.

The simulation results will be extended to real experiments, leveraging the scalability and adaptability of the Active Inference controller [109]. The steady-state errors (e_{ss}) and the $RMSE$ between beliefs and actual joint positions for the simulations are reported in Table 3.1.
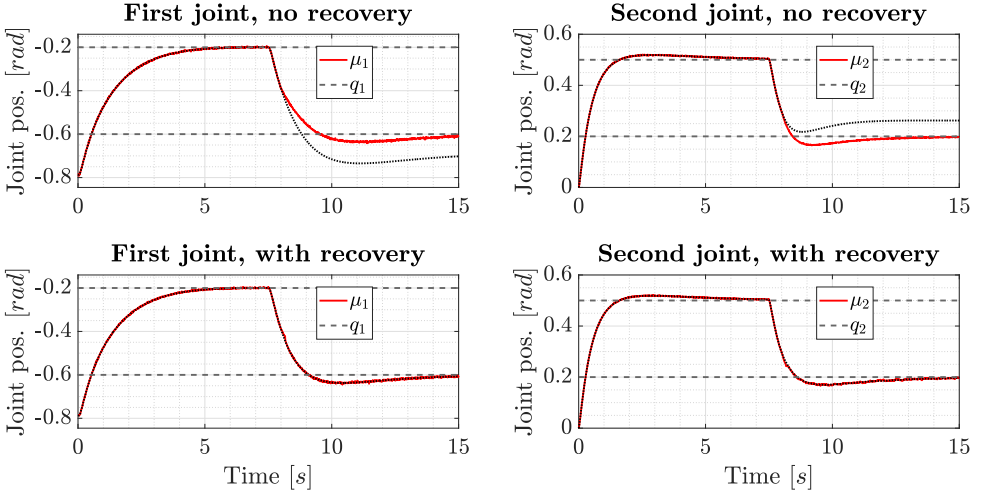
Figure 3.5: System's response in case of encoder fault without recovery (fixed precision matrices) and with recovery (setting $P_{y_q} = \mathbf{0}$ after detection). The fault occurs after 8s.
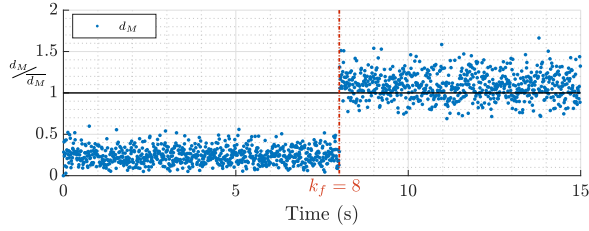
Figure 3.6: Normalised d_M in case of camera fault.

Table 3.1: Summary of the results using u-AIC

Metrics	Encoder Fault		Camera Fault	
	Recovery	No recovery	Recovery	No recovery
$e_{ss,\ q1}$	$-2.7e^{-3}$	0.1674	$-2.1e^{-4}$	0.0173
$e_{ss,\ q2}$	$-1.5e^{-3}$	-0.1176	$-2.5e^{-4}$	$7.0e^{-3}$
$RMSE_{q_1}$	$3.0e^{-3}$	0.0896	$5.7e^{-4}$	0.0112
$RMSE_{q_2}$	$2.5e^{-3}$	0.0636	$3.5e^{-4}$	$2.1e^{-3}$

3.7. Conclusions

In this paper, we presented a new formulation of an Active Inference controller where the free-energy depends explicitly on the control actions. This formulation brings two general advantages. First, it allows us to remove the bias from state estimation, which affected the standard Active Inference controller, leading to more accurate state estimation. Second, it simplifies the definition of the control actions since it

removes the need to compute partial derivatives of the free-energy. Next, we showed how the new formulation simplifies the use of established fault detection techniques with probabilistic robustness, which would have caused many false positives using standard Active Inference. The effectiveness of the approach is showcased in a simulated 2-DOF arm subject to sensory faults. Future work will investigate fault-tolerant control with stochastic decision boundaries, a formal stability proof, and experiments on a real robot manipulator.

3

4

Unbiased Active Inference for Classical Control

*I*N this chapter, we present a formalization of the concept of unbiased Active Infer-
ence, which was initially introduced in Chapter 3. Taking a mathematical stand-
point, we establish a comprehensive framework that extends beyond its application
in fault-tolerant control. In fact, we examine the limitations of the AIC not only
in case of faults but also in reference tracking and collision scenarios. To validate
our findings, we conduct experiments on a real 7-DOF robot arm, marking the first
implementation of unbiased Active Inference on an actual robot.

Figure 4.1: Manipulation of delicate items in a human-shared store environment.

4.1. Abstract

Active inference is a mathematical framework that originated in computational neuroscience. Recently, it has been demonstrated as a promising approach for constructing goal-driven behavior in robotics. Specifically, the Active Inference controller (AIC) has been successful on several continuous control and state-estimation tasks. Despite its relative success, some established design choices lead to a number of practical limitations for robot control. These include having a biased estimate of the state, and only an implicit model of control actions. In this paper, we highlight these limitations and propose an extended version of the unbiased Active Inference controller (u-AIC). The u-AIC maintains all the compelling benefits of the AIC and removes its limitations. Simulation results on a 2-DOF arm and experiments on a real 7-DOF manipulator show the improved performance of the u-AIC with respect to the standard AIC. The code can be found at https://github.com/cpezzato/unbiased_aic.

4.2. Introduction

Active inference is a mathematical framework prominent in computational neuroscience [47]. It aims to explain decision-making in biological agents using free-energy minimization. Recent work has led to many schemes based on this framework; for an overview, see the review paper from Lanillos and coauthors [78]. Nevertheless, Active Inference for robot control is still a relatively new field. It is thus of particular importance to understand the limits of such methods. In this paper, we analyze the drawbacks associated with specific design choices within the Active Inference controller (AIC). Additionally, we propose an extended and improved unbiased AIC (u-AIC), which we initially presented in a previous publication [9] solely for fault-tolerant control. The effort to understand the limits of the AIC for robot control started with our previous work [9], where we specifically analyzed the effect of joint

state estimation and control for fault tolerance. The current paper provides an in-depth analysis of the performance and limits of the AIC more generally (not just fault-tolerant control). Additionally, we formalize the u-AIC. This includes 1) a derivation of the new control architecture and possible extensions, 2) a proof of convergence for both the state estimation and the controlled system, and 3) the application of the u-AIC to a real 7-DOF robot manipulator, which was previously only performed in simulation.

In this paper, we identify the drawbacks of the AIC and connect them to two root causes. First, in the AIC the estimated state (or *belief*) is biased toward the current goal/target state through a goal prior. This leads to reduced quality in state-estimation [6] but also influences precision learning (learning the precision/covariance of the sensory model), making the model parameters converge to a biased value [6, 11]. A range of issues also arises in fault diagnosis and fault-tolerant control as a result of the goal prior, such as false-positive fault detection [107, 9]. Second, the control action is not explicitly modeled as a random variable in the generative model of the AIC. This causes a range of issues on the control side. In real-world control applications, the integral control law typical of the AIC can cause saturation problems for the actuators when the agent fails to reach a target. This can happen, for instance, because of a collision. Other limitations from a control perspective are that the AIC does not naturally allow for the incorporation of feed-forward control signals [9], which could improve the overall performance of the system. Finally, the motion can be jerky in practice.

Interestingly, all these limitations are present in the AIC but are not intrinsically part of Active Inference as a general framework. One can design different controllers that stay true to the principles of Active Inference while mitigating the limitations. To this end, we present the u-AIC, previously introduced in our previous paper [9]. We demonstrate the properties of the u-AIC in Section 4.5, as well as the convergence of both the state estimation and control, which is still missing for the AIC [109].

The *contributions* of this paper are twofold: 1) we concretely demonstrate the limitations of the AIC using both theoretical results and empirical demonstrations. 2) We formalize the u-AIC and show how it overcomes such limitations, providing proof of convergence and extensions to the controller. We believe that understanding the properties and the boundaries of the AIC is important for researchers who want to apply it for robot control, such that better and safer systems can be built using this framework.

4.2.1. Related work

Active inference has been proposed in neuroscience as a general theory of the brain [45] and was later concisely reported using a notation and language closer to engineering and control [21]. We refer to the controller in this notation as AIC. From this, the first real-world applications for robot control [109, 98] were derived. Recently, Active Inference has been applied for robot control in a broader variety of settings and tasks. This includes work on robust state-estimation [77, 88], adaptive control [109, 6], fault-tolerant control [9, 8], reinforcement learning [135], planning

under uncertainty [35, 7], human-robot interaction [28, 97] and more. A recent survey [78] provides a detailed overview of Active Inference for robot control to date.

Particularly interesting in the context of this paper, is that several recent approaches in robotics have taken inspiration from the free-energy principle and applied it to continuous control and state estimation. Recent works focused on chance-constrained Active Inference [137], LQG control [76], and model-predictive control [5]. However, while the area of application of Active Inference in engineering settings is continuously growing, works on in-depth analysis of the AIC from a control theoretic perspective are limited. Previous work [12, 6] has shown the relationship between the AIC and PID controllers, while the relationship between the AIC, LQR control, and Kalman filters is discussed in other publications [13, 76]. Finally, we discussed some limitations of the AIC in the context of fault-tolerant control [107, 9]. This motivated the introduction of the u-AIC [9]. This work focuses on a thorough analysis of the limits of the AIC from a control point of view and proposes an extended version of the u-AIC to address them.

4.2.2. Structure of the paper

The remainder of this paper is organized as follows. After providing the necessary mathematical background in Section 4.3, Section 4.4 details the limitations of the AIC when applied to robot control. Section 4.5 presents the u-AIC scheme to overcome these limitations. In Section 4.6, the properties of the u-AIC and its relation with the AIC are highlighted. In Section 4.7, the theoretical claims are validated in simulation and on a real 7-DOF Franka Emika Panda manipulator. Finally, we draw conclusions in Section 4.8.

4.3. Preliminaries

In this section, we concisely report the AIC formulation as used in previous work [109, 6, 89, 98], which specified the AIC for robot control according to the theory [21]. In this section, we only report the crucial equations to understand the limitations of the AIC in Section 4.4; an interested reader is referred to past work for all the details [109].

4.3.1. The generative model

Active Inference considers an agent in a dynamic environment that receives observations $\boldsymbol{y}$ about states $\boldsymbol{x}$ at every time-step t. The generative model of the agent can then be expressed as:

$$p(\boldsymbol{x}, \boldsymbol{y}) = \underbrace{p(\boldsymbol{y}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior}. \tag{4.1}$$

A visual representation of this model is presented in fig. 4.2 (right). The probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ has a mean $\boldsymbol{g}(\boldsymbol{x})$, which is a mapping from state to observation. The prior $p(\boldsymbol{x})$ has a mean $\boldsymbol{f}(\boldsymbol{x})$, which is a function that encodes the goal state $\boldsymbol{\mu}_g$. The agent aims to infer the posterior $p(\boldsymbol{x}|\boldsymbol{y})$ given a model of the

agent's world. This means finding the belief $\boldsymbol{\mu}$ over the state $\boldsymbol{x}$ given the observations. This can be achieved by minimizing the so-called variational free-energy. If all distributions in eq. (4.1) are Gaussian, the free-energy becomes a sum of least square terms [109].

4.3.2. Free-energy

The AIC performs joint state estimation and control by minimizing the free-energy $\mathcal{F}$ through a gradient descent scheme. $\mathcal{F}$ is defined as [109]:

$$\mathcal{F}(\boldsymbol{y}, \boldsymbol{\mu}) = \frac{1}{2} \sum_{i=0}^{n_d-1} \left[\boldsymbol{\varepsilon}_y^{(i)\top} \Sigma_{y^{(i)}}^{-1} \boldsymbol{\varepsilon}_y^{(i)} + \boldsymbol{\varepsilon}_\mu^{(i)\top} \Sigma_{\mu^{(i)}}^{-1} \boldsymbol{\varepsilon}_\mu^{(i)} \right] + K. \tag{4.2}$$

The free-energy is a weighted sum of prediction errors up to a constant K resulting from the derivations [21]. The terms $\Sigma_{y^{(i)}}^{-1}$ and $\Sigma_{\mu^{(i)}}^{-1}$ are precision matrices representing the confidence about sensory input and internal beliefs. These can be seen as tuning parameters. The term n_d represents the number of derivatives considered in the control problem. If we assume, as in most cases [109, 98, 9] that $n_d = 2$, then state estimation and control are performed on position and velocity. These quantities are internally represented as the beliefs $\boldsymbol{\mu}^{(0)} = \boldsymbol{\mu}$ for positions, and $\boldsymbol{\mu}^{(1)} = \boldsymbol{\mu}'$ for velocities. The terms $\boldsymbol{\varepsilon}_\mu^{(i)} = (\boldsymbol{\mu}^{(i+1)} - \boldsymbol{f}^{(i)}(\boldsymbol{\mu}))$ and $\boldsymbol{\varepsilon}_y^{(i)} = (\boldsymbol{y}^{(i)} - \boldsymbol{g}^{(i)}(\boldsymbol{\mu}))$ are respectively the *state* and *sensory* prediction errors.

The function $\boldsymbol{g}(\boldsymbol{\mu})$ represents the mapping between states and sensory observations. In the case of robot control with position and velocity sensors, this is the identity mapping, so $\boldsymbol{g}(\boldsymbol{\mu}) = \boldsymbol{\mu}$. The function $\boldsymbol{f}(\boldsymbol{\mu}) = \boldsymbol{\mu}_g - \boldsymbol{\mu}$ specifies the desired evolution of the dynamics of the system. In this case, [109], the AIC will make the system behave like a first-order linear system with the desired goal position $\boldsymbol{\mu}_g$. By definition [21, 109], it holds:

$$\boldsymbol{g}^{(i)} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)}, \ \boldsymbol{f}^{(i)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)}, \ \boldsymbol{g}^{(0)} = \boldsymbol{g}, \ \boldsymbol{f}^{(0)} = \boldsymbol{f}. \tag{4.3}$$

Finally, the terms $\Sigma_{\mu^{(i)}}$ and $\Sigma_{y^{(i)}}$ are diagonal covariance matrices. In the scalar case, these are represented as the variances σ_μ and σ_y.

4.3.3. State estimation

State estimation in the AIC is achieved by gradient descent on the free-energy [21, 52, 109] with the update rule:

$$\dot{\tilde{\boldsymbol{\mu}}} = \frac{d}{dt}\tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{\mu}}}, \tag{4.4}$$

where $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}, \ \boldsymbol{\mu}']$, and t refers to time. The term κ_μ is a tunable learning rate.

4.3.4. Control

In the AIC, the control input $\boldsymbol{u}$ is also computed through gradient descent on $\mathcal{F}$. As can be seen by eq. (4.2), however, $\mathcal{F}$ does not depend on $\boldsymbol{u}$ explicitly. On the

other hand, the actions indirectly influence $\mathcal{F}$ by making the state evolve over time and thus changing the sensory output. We can compute the actions using the chain rule as [21, 109]:

$$\dot{\boldsymbol{u}} = -\kappa_a \frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{u}} \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{y}}} \tag{4.5}$$

where $\tilde{\boldsymbol{y}} = [\boldsymbol{y}, \ \boldsymbol{y}']$, and κ_a is the tuning parameter. The term $\frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{u}}$, requires a forward dynamic model and is generally hard to compute in closed-form for non-linear systems. In most previous work [109, 98, 9, 6], such need has been obviated by introducing a linear approximation. The expression can alternatively be written as a sum for every sensor $\boldsymbol{y}$ in $\tilde{\boldsymbol{y}}$. This relationship shows how the control action is directly related to the sensory inputs and the *number* of sensors, as seen in the following expression:

$$\dot{\boldsymbol{u}} \approx -\kappa_a \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{y}}} = -\kappa_a \sum_y \frac{\partial \mathcal{F}}{\partial \boldsymbol{y}}. \tag{4.6}$$

As an example, for a system with a position sensor $\boldsymbol{y_q}$, and velocity sensor $\boldsymbol{y_{\dot{q}}}$ this control law would be:

$$\dot{\boldsymbol{u}} = -\kappa_a \left[\Sigma_{y_q}^{-1} (\boldsymbol{y_q} - \boldsymbol{\mu}) + \Sigma_{y_{\dot{q}}}^{-1} (\boldsymbol{y_{\dot{q}}} - \boldsymbol{\mu}') \right]. \tag{4.7}$$

4.4. Limitations of the AIC

In this section, we discuss the limitations of the AIC for robot control. We note that these limitations are not inherent in the Active Inference framework but rather in how the AIC is constructed. In Section 4.4.1 we address *Limitation #1*: in AIC the belief over the *current* state is biased toward the target the agent aims to reach by means of goal prior. This means that the agent's belief is never accurate, except when the target is reached. In Section 4.4.2 we address *Limitation #2*: the control action is not explicit in the generative model. This means, that one cannot minimize the free-energy with respect to the actions directly and instead use the chain rule as in eq. (4.5), making assumptions about the linearity of the system. This can cause a saturation problem and does not allow for the incorporation of feed-forward control signals to improve performance.

4.4.1. Limitation #1: biased state estimation

Before formally analyzing the controller, we provide a visual explanation using factor graphs. In fig. 4.2, the generative model of the AIC in eq. (4.1) is depicted. Each random variable is represented by a circle, and each probability distribution by a black square. By inspecting this graph, we see that the state $\boldsymbol{x}$ is connected to two distributions. The distribution $p(\boldsymbol{y}|\boldsymbol{x})$ moves the belief closer to the observed value $\boldsymbol{y}$. This distribution is represented in the free-energy $\mathcal{F}$ by the term $\boldsymbol{\varepsilon}_y^\top \Sigma_y^{-1} \boldsymbol{\varepsilon}_y$ where $\boldsymbol{\varepsilon}_y = (\boldsymbol{y} - \boldsymbol{\mu})$. This is considering, as commonly done, $\boldsymbol{g}(\cdot)$ as the identity mapping,

The second distribution is the prior $p(\boldsymbol{x})$. This distribution is Gaussian with the function $\boldsymbol{f}(\boldsymbol{\mu}) = \boldsymbol{\mu}_g - \boldsymbol{\mu}$ as its mean. The function encodes the goal state $\boldsymbol{\mu}_g$. This term moves the belief $\boldsymbol{\mu}$ towards the goal state. The belief of the AIC is thus always

biased towards the goal state. This is by design. The benefit of this is that we can now compute a control action that moves the agent to the goal (using eq. (4.6)). However, the drawback is that state estimation is inaccurate.
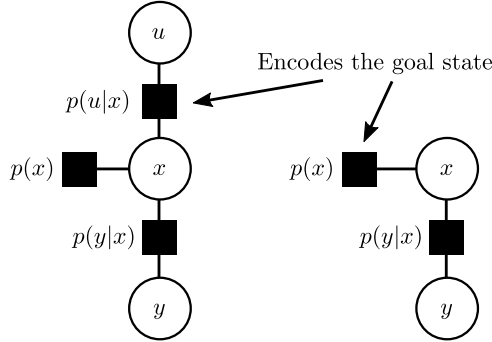
Figure 4.2: Illustration of the u-AIC (left) and the AIC (right) for a one-dimensional problem. Circles indicate random variables. Black boxes indicate probability distributions.

Convergence to a biased belief
We will now analyze the convergence of the beliefs in the AIC. Let us consider a generic form of the free-energy, as in eq. (4.2). For the simplest linear and scalar case with $n_d = 1$ (i.e. one sensor and one actuator, with observable state), this expression can be reduced to:

$$\mathcal{F} = \frac{1}{2}\left[\frac{(y - g(\mu))^2}{\sigma_y} + \frac{(\mu' - f(\mu))^2}{\sigma_\mu}\right] + K \tag{4.8}$$

Let us consider the generative model of the state dynamics as a first-order linear system with unitary time constant, so $f(\mu) = \mu_g - \mu$, and since the state is observable, so $g(\mu) = \mu$ [109]. At a steady state, it holds that

$$\frac{\partial \mathcal{F}}{\partial \mu} = -\frac{(y - \mu)}{\sigma_y} + \frac{(\mu' - \mu_g + \mu)}{\sigma_\mu} = 0. \tag{4.9}$$

Additionally, when solving for μ we can show that

$$\mu = \frac{\sigma_\mu y + \sigma_y \mu_g - \sigma_y \mu'}{\sigma_\mu + \sigma_y}. \tag{4.10}$$

From eq. (4.10), one can notice that the belief μ is a weighted sum of the sensory measurement and the goal state. The belief is thus always biased towards the goal. We demonstrate incorrect state estimation in simulation (section 4.7.1), which is particularly evident when collisions happen with the environment. The bias in the state estimation leads to two other issues. These are apparent when one tries to optimize model parameters through *precision learning* (or inverse covariance learning), and when applying the AIC for *fault-tolerant control*. We expand upon these claims in the following sections.

Biased precision learning

Past work on robot control [109, 98] assumed that the precision σ_y^{-1} of the observation model is known. Intuitively, this quantifies the confidence about how noisy the sensor is and can be determined before the deployment of the controller. However, for certain applications where sensory noise is uncertain, one might need to estimate the precision of the sensor online. This is the case in many robotics applications [138, 111].

There exists a body of work related to Active Inference that explores precision learning., e.g. methods such as Dynamic Expectation Maximization and generalized filtering [43, 52]. However, these methods do not perform control. When precision learning is done in conjunction with the Active Inference controller, we cannot estimate the true precision of the sensor. Previous works [11, 6] show how precision learning in the AIC can be used to find the optimal gains of the controller. In this context, however, the obtained precision of the sensor loses its physical meaning.

As explained earlier, the belief over the current state is biased, and this propagates to the precision of the system. Consider again the simplest linear and scalar case of the free-energy in eq. (4.8). The constant K can be expanded to include the variances as [21]:

$$K = \frac{1}{2} \ln \sigma_y \sigma_\mu.$$

Considering these terms time-varying, we can take the gradient with respect to the variances as:

$$\frac{\partial \mathcal{F}}{\partial \sigma_y} = -\frac{(y - \mu)^2}{2\sigma_y^2} + \frac{1}{2\sigma_y}. \tag{4.11}$$

Setting this expression to zero, we obtain:

$$\sigma_y = (y - \mu)^2. \tag{4.12}$$

The updated variance depends on μ, which we showed is biased. Thus the precision will also be biased. This is because the agent is estimating the precision as the average square distance between the mean μ and every measurement y. If μ is biased, then the agent is estimating the variance around a value that is not the true mean. A special case is when the agent already starts at the goal. In that setting, μ would represent the true mean, and the precision would be estimated around the true mean resulting in an accurate precision estimate. Finally, learning the precision of the observation model can be used in the context of control, *but* the obtained precision no longer represents the noise level of the sensor. Instead, it is a combination of the noise level, and how aggressive the controller will act, as shown in previous work [5].

Limitation in fault-tolerant control

In previous work, we used the AIC for fault-tolerant control [107] where the main idea was that the sensory prediction errors in the free-energy could be used as residuals for fault detection, removing the need for more advanced residual generators. Despite the simplicity of the method for detecting and recovering from broken sensors, the use of prediction errors with biased state estimation can cause several false

positives. This is explained in detail in previous work [9], where a first version of the u-AIC has been proposed.

4.4.2. Limitation #2: implicit modelling of actions

The control law for the AIC is computed using gradient descent on $\mathcal{F}$, as in eq. (4.5). Since the action is not explicitly modeled, one resorts to the chain rule. This introduces additional complexity. The term $d\boldsymbol{y}/d\boldsymbol{\mu}$ is generally hard to compute, and researchers approximated it by the identity matrix [11, 109]. Despite the promising results, linearizing the forward dynamics necessarily limits the performance in the presence of non-linear dynamics. Additionally, the control law is driven by sensory prediction errors weighted by the precision, see eq. (4.7), and it is de-facto an integrator. This causes three issues in practice.

First, the control law in the AIC is directly dependent on the observation. Eq. (4.6) shows that the control law is a sum of sensory prediction errors. This means that, if a system is not observable, it is not controllable. More specifically, past works [6, 11] detail how, under certain assumptions, the Active Inference controller is equivalent to a PI Controller i.e. PID with $D = 0$, a P gain of $\kappa_u \Sigma_{y'}^{-1}$ and an I gain of $\kappa_u \Sigma_y^{-1}$. This assume that the function $\boldsymbol{f}(\boldsymbol{\mu}) = (\boldsymbol{\mu}_g - \boldsymbol{\mu})\tau^{-1}$ has a $\tau \to 0$. In that case, the belief will approach the goal state $\boldsymbol{\mu} \to \boldsymbol{\mu}_g$. In eq. (4.6), we see that the control law of the AIC is the sum of sensory prediction errors. Every term, in this case, is responsible for an error term in the PID control law. A position sensor is responsible for the I gain, a velocity sensor will result in a P gain, and an acceleration sensor will result in a D gain.

Second, integral control has a few general drawbacks. For instance, if the goal cannot be reached due to a collision, the magnitude of the control action $\boldsymbol{u}$ will monotonically increase until saturation. This is shown in section 4.7 for both simulated and real robots, where we point out that a vanilla implementation of the AIC leads to integrator windup.

Third, the control action cannot be naturally supplemented with a feed-forward signal. If one has information about the system, designing a feed-forward signal to supplement the feedback controller consistently improves performance.

4.5. Unbiased Active Inference controller

A first version of the u-AIC has been introduced in previous work [9] in the context of fault-tolerant control. In this section, we generalize, extend, and formally analyze the previously proposed u-AIC for generic control settings, and show how it overcomes the limitations of the AIC.

4.5.1. Derivation of the u-AIC

In this section, we describe the u-AIC as introduced in past work [9], to which an interested reader is referred for more details on the derivations of the following equations. Let us consider $\boldsymbol{x} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]^\top$ and let us define a probabilistic model where

actions are modeled explicitly:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \quad \underbrace{p(\boldsymbol{y}|\boldsymbol{x})}_{observation\ model} \quad \underbrace{p(\boldsymbol{x})}_{prior} \tag{4.13}$$

A visual representation of the u-AIC can be seen in fig. 4.2 (left). Note that with the u-AIC the information about the desired goal to be reached is encoded in the distribution $p(\boldsymbol{u}|\boldsymbol{x})$. This fundamentally removed the bias of the current state, as will be shown. In this paper, as in past work [6], we assume that an accurate dynamic model of the system is not available to keep the solution system agnostic and to highlight once again the adaptability of the controller.

The u-AIC aims at finding the posterior over states as well as the posterior over actions $p(\boldsymbol{x}, \boldsymbol{u}|\boldsymbol{y})$. Since the posteriors can be difficult to compute exactly, they are approximated using a variational distribution $Q(\boldsymbol{x}, \boldsymbol{u})$. We can make use of the mean-field assumption $(Q(\boldsymbol{x}, \boldsymbol{u}) = Q(\boldsymbol{x})Q(\boldsymbol{u}))$ and the Laplace approximation, and assume the posterior over the state $\boldsymbol{x}$ is Gaussian with mean $\boldsymbol{\mu}_x$ [49]. Similarly, for the actions, the posterior $\boldsymbol{u}$ is assumed Gaussian with mean $\boldsymbol{\mu}_u$. By defining the Kullback-Leibler divergence between the variational distribution and the true posterior, one can derive an expression for the free-energy $\mathcal{F}$ as [9]:

$$\mathcal{F} = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}) + C \tag{4.14}$$

Considering eq. (4.13) and assuming Gaussian distributions, $\mathcal{F}$ becomes:

$$\begin{aligned} \mathcal{F} = \frac{1}{2}(\boldsymbol{\varepsilon}_y^\top \Sigma_y^{-1} \boldsymbol{\varepsilon}_y + \boldsymbol{\varepsilon}_x^\top \Sigma_x^{-1} \boldsymbol{\varepsilon}_x \\ + \boldsymbol{\varepsilon}_u^\top \Sigma_u^{-1} \boldsymbol{\varepsilon}_u + \ln|\Sigma_u \Sigma_y \Sigma_x|) + C, \end{aligned} \tag{4.15}$$

The terms $\boldsymbol{\varepsilon}_{y_q} = \boldsymbol{y}_q - \boldsymbol{\mu}$, $\boldsymbol{\varepsilon}_{y_{\dot{q}}} = \boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}'$ are the sensory prediction errors respectively for position and velocity sensory inputs. The controller represents the states internally as $\boldsymbol{\mu}_x = [\boldsymbol{\mu}, \boldsymbol{\mu}']^\top$. The relation between internal state and observation is expressed through the generative model of the sensory input $\boldsymbol{g} = [\boldsymbol{g}_q, \boldsymbol{g}_{\dot{q}}]$. Position and velocity encoders directly measure the state, thus, $\boldsymbol{g}_q$ and $\boldsymbol{g}_{\dot{q}}$ are linear (identity) mappings.

Additionally, $\boldsymbol{\varepsilon}_u$ is the prediction error on the control action, while $\boldsymbol{\varepsilon}_x$ is the prediction error on the state. The latter is computed considering a prediction of the state $\hat{\boldsymbol{x}}$ at the current time-step such that $\boldsymbol{\varepsilon}_x = (\boldsymbol{\mu}_x - \hat{\boldsymbol{x}})$. The prediction is a deterministic value $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}]^\top$ which can be computed in the same fashion as the prediction step of, for instance, a Kalman filter. The prediction is approximated by propagating forward in time the current state belief using the following simplified discrete time model:

$$\hat{\boldsymbol{x}}_{k+1} = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \boldsymbol{\mu}_{x,k} \tag{4.16}$$

where I represents a unitary matrix of suitable size. This form assumes that the position of each joint is thus computed as the discrete-time integral of the velocity, using a first-order Euler scheme. This approximation can be avoided if a better

dynamic model of the system is available, and in that case, predictions can be made using the model itself. Finally, by choosing the distribution $p(\boldsymbol{u}|\boldsymbol{x})$ to be Gaussian with mean $\boldsymbol{f}^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_g)$, we can steer the system towards the target $\boldsymbol{\mu}_g$ without biasing the state estimation. This results in $\boldsymbol{\varepsilon}_u = (\boldsymbol{\mu}_u - \boldsymbol{f}^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_g))$.

In the u-AIC state estimation and control are achieved using gradient descent along the free-energy. This leads to:

$$\dot{\boldsymbol{\mu}}_u = -\kappa_u \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_u}, \quad \dot{\boldsymbol{\mu}}_x = -\kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_x}, \tag{4.17}$$

where κ_u and κ_μ are the gradient descent step sizes. The gradient on control can be computed as:

$$\frac{\partial F}{\partial \boldsymbol{\mu}_u} = \Sigma_u^{-1}(\boldsymbol{\mu}_u - \boldsymbol{f}^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_g)) \tag{4.18}$$

4.5.2. Proof of convergence
For the simple 1-dimensional case, the expression of the free-energy for the u-AIC can be reduced to:

$$\mathcal{F} = \frac{1}{2}\left[\frac{(y - \mu_x)^2}{\sigma_y} + \frac{(\mu_u - f^*(\cdot))^2}{\sigma_u} + \frac{(\mu_x - \hat{x})^2}{\sigma_x}\right] + K \tag{4.19}$$

The belief over the state $\boldsymbol{\mu}_x$ and over the control action $\boldsymbol{\mu}_u$ will converge when:

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_u} = 0, \quad \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_x} = 0, \tag{4.20}$$

At steady state, it holds that $\mu_u = f^*(\mu_x, \mu_g)$. Considering this result, one can show that μ_x converges to

$$\mu_x = \frac{\sigma_x y + \sigma_y \hat{x}}{\sigma_x + \sigma_y}. \tag{4.21}$$

We can thus see that the control action converges to the function f^*, which can be chosen, for instance, as a PID controller. We can also show that belief over the state is a weighted average of the sensory measurement y and the prediction $\hat{x}$. Unlike the AIC, this prediction does not depend on the goal, see eq. (4.10).

4.5.3. Extensions of the u-AIC for control
The u-AIC can be extended for richer control by modifying the probabilistic model in eq. (4.13). This is because the control action $\boldsymbol{u}$ is explicitly modeled as a random variable, and thus we can add prior probability distributions to it. This can be done in multiple ways to achieve different objectives.

Adding a feed-forward controller (open-loop)
So far, the only term depending on the control action now is $p(\boldsymbol{u}|\boldsymbol{x})$; however, a prior $p(\boldsymbol{u})$ can also be added. In that case, the generative model would be:

$$\frac{1}{\alpha}p(\boldsymbol{u})p(\boldsymbol{u}|\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \tag{4.22}$$

where α is a normalization constant. Note that, this denominator does not need to be computed explicitly. To achieve state estimation and control, we need to minimize the free-energy (which maximizes the likelihood of the model). We do not need to compute the free-energy or the likelihood exactly.

This prior can encode a feed-forward signal (open-loop control law). Assuming the prior to be Gaussian with mean $f_{ol}(x)$ and variance σ_{ol} ('ol' stands for open-loop), we can show that the control law will converge to:

$$\mu_u = \frac{f_{ol}(\mu_x)\sigma_u + f^*(\mu_g, \mu_x)\sigma_{ol}}{\sigma_{ol} + \sigma_u}. \tag{4.23}$$

This is the weighted sum of the PID control law (after applying a filter) and the open-loop control law. Note previously, the value of σ_u did not matter. Now the ratio between σ_u and σ_{ol} does contribute. Additionally, the expression for $\mathcal{F}$, would contain a quadratic term for the open-loop control law.

Adding control costs
Alternative to adding a feed-forward control law, we might add a control cost. This can be achieved by adding the control prior $p_{cc}(\boldsymbol{u})$ with a mean of 0 and variance of σ_{cc}. This creates a quadratic term in $\mathcal{F}$ of $(\mu_u - 0)^2/\sigma_{cc}$. This is equivalent to a quadratic control cost, as seen in classical LQR controllers.

Smoothing the control action
The AIC often exhibits jerky motion. This can be mitigated in the u-AIC by adding a smoothing prior. Let u be a random variable referring to the current control action being executed. We then define u_p as the control action executed at the time previous time-step. A distribution $p(u|u_p)$ can be added to the generative model. This will add a control cost of $(\mu_u - u_p)^2/\sigma_p$ to $\mathcal{F}$. Minimizing this quantity nudges the control action u toward the value of the last control action u_p, creating a smoothing effect. This quadratic loss is similar to approaches in Model-predictive control (MPC) [99].

A comparison of the standard AIC and u-AIC against MPC and impedance control can be found in the literature [89]. Future work could address the comparison of the proposed extensions of the u-AIC against other classical controllers.

4.6. Relationship between the AIC and u-AIC

4.6.1. Convergence of beliefs
The AIC considers the relationship between states and observations without explicitly modeling the control actions. In classical filters, such as the Kalman filter, the prior comes from a prediction step that relies on the previous state and action. In the case of the AIC, the prior essentially predicts the agent will move towards the target. This results in a biased state estimate as seen in eq. (4.10). In contrast, the u-AIC has an unbiased belief over the state as seen in eq. (4.21). Note that both expressions are almost identical. The AIC converges to the weighted average between the sensory measurement and the goal. The u-AIC, on the other hand, converges to a weighted average between the sensory measurement and the predicted

state (using a model or Euler integration). Note that, in the AIC, the **goal** state is encoded in the prior over the state $p(\boldsymbol{x})$, while in the u-AIC it is encoded in a separate distribution $p(\boldsymbol{u}|\boldsymbol{x})$ (see fig. 4.2).

4.6.2. Control law

In the u-AIC, actions are explicitly modeled and the target state is encoded in the term $p(\boldsymbol{u}|\boldsymbol{x})$. Explicit actions allow us to directly perform gradient descent on $\mathcal{F}$ with respect to $\boldsymbol{u}$. This is not possible in the general AIC case and the chain rule had to be utilized (see eq. (4.5)). The eventual control law of the AIC is also directly dependent on the measurement and the number of measurements eq. (4.6). Thus if part of the system is unobserved, it can not be controlled. The u-AIC on the contrary has a control law irrespective of the number of sensors eq. (4.21) and allows us to encode control costs, smoothing effects, and feed-forward control signals (see eq. (4.23)).

4.6.3. General architecture

The general architecture (for state estimation and control) for the AIC and u-AIC is also different. In the case of the AIC, state estimation is dependent on the goal state. The goal is encoded in the prior, which biases the state estimation. The controller, on the other hand, is dependent on the (biased) belief and measurements. This is unusual in classical control, see fig. 4.3. A discussion on the role of modularity in AIC and the general control architecture can be found in the work from Baltieri and Buckley [14]. The u-AIC on the other hand has a more traditional flow of

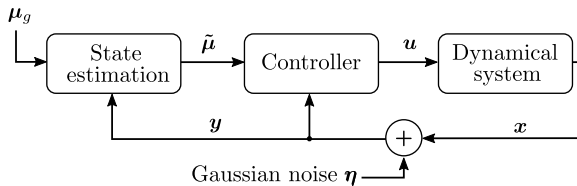
Figure 4.3: Control diagram of the AIC

information. State estimation requires the measurements $\boldsymbol{y}$. Then the (unbiased) belief $\boldsymbol{\mu}_x$ and the goal $\boldsymbol{\mu}_g$ are fed into the controller which produces the control action. This is illustrated in fig. 4.4.

Figure 4.4: Control diagram of the u-AIC.

(a) No collision   (b) Collision
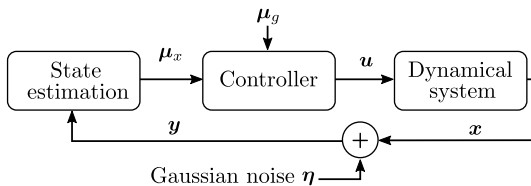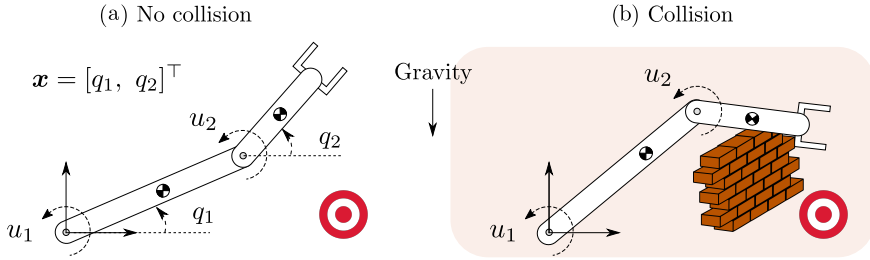
Figure 4.5: Scenarios considered to illustrate the incorrect state estimation due to the bias towards the target (red dot). In (b) an obstacle occludes the way and blocks the arm.

4

4.7. Experimental evaluation

We now showcase the limitations of the AIC explained in Section 4.4 and compare the performance with the u-AIC, both in simulation and in the real world.

4.7.1. Simulation

The simulation scenario is depicted in fig. 4.5. The robot has to reach a target in configuration space. During motion, we suppose that at a certain time, a collision occurs which prevents the robot from moving further. We assume the collision persists for $t_c = 3s$, then the robot is free to proceed. This allows us to show the incorrect state estimation and the overshoot due to the integral control law of the AIC. In the u-AIC, we observe none of these while maintaining similar performance. The 2-DOF robot arm is equipped with position and velocity sensors $\boldsymbol{y}_q$, $\boldsymbol{y}_{\dot{q}} \in \mathbb{R}^2$ for the two joints affected by zero mean Gaussian noise, as in fig. 4.5. We define $\boldsymbol{y} = [\boldsymbol{y}_q, \ \boldsymbol{y}_{\dot{q}}]^\top$. The states $\boldsymbol{x}$ to be controlled are set as the joint positions $\boldsymbol{q} = [q_1, q_2]^\top$ of the robot arm. The simulation results for AIC and u-AIC are reported in fig. 4.6. The collision scenario in fig. 4.5 highlights a few limitations:

Incorrect state estimation

Let us consider the first row of the plots in fig. 4.6. When the AIC cannot reach the desired target due to a collision blocking the path, the belief $\boldsymbol{\mu}$ does not follow the trend of the real state $\boldsymbol{x}$. The belief converges to a value between the sensory reading and the desired goal. For instance, at time $t = 4s$ the sensory reading is $-0.355 \ [rad]$. Given the current $\mu_g = -0.2 \ [rad]$ as goal for the first joint, $\sigma_y = \sigma_\mu = 1$, and $\mu' = 0.053 \ [rad/s]$ the belief converges to $-0.304 \ [rad]$, which is in accordance with eq. (4.10). On the other hand, the u-AIC converges to the true state. For statistical significance, we ran 100 reaching tasks for each controller, with random blocking collisions of duration between 1 and 3 $[s]$ at a time between 0 and 3 $[s]$. Table 4.1 reports steady-state error (e_{ss}), settling time (t_s) after removing the collision, overshoot (os), and $RMSE$ between beliefs and joint positions in case of blocking the arm temporarily, averaged over the 100 trials. As can be seen, while the AIC presents the lowest e_{ss} due to the prominent integration scheme, the $RMSE$ for state estimation is two orders of magnitude higher than for the u-AIC. Incorrect state estimation can also cause several false positives [9]. The sensory prediction
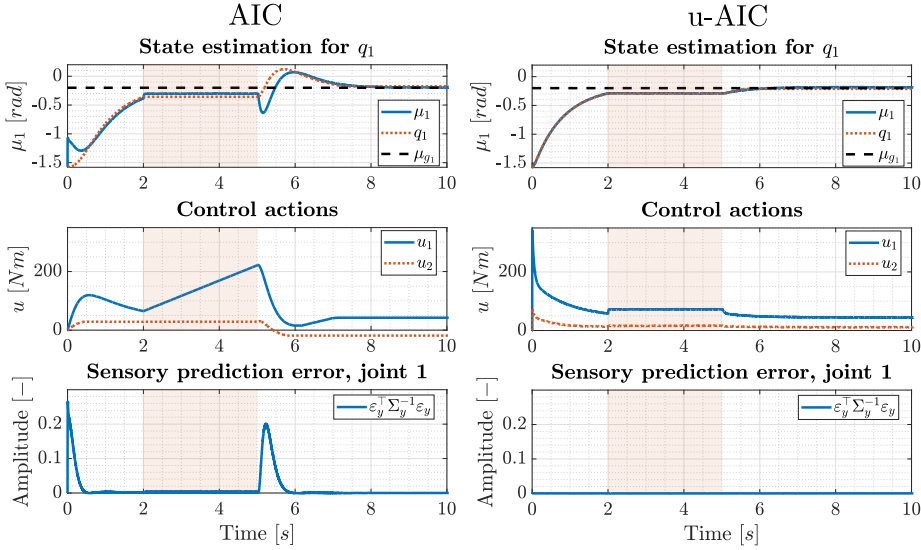
Figure 4.6: Simulation results of a 2-DOF robot arm collision (orange area) for $t_c = 3s$. For readability, we only report the first joint since the behavior is similar to the second one.

errors for the AIC can, in fact, increase due to collisions with the environment (see the last row of plots in fig. 4.6). The u-AIC is instead insensitive.

Scenarios	e_{ss} [rad]	t_s [s]	os [%]	RMSE [rad]
AIC	2.82e-05	3.70	0.12	0.042
u-AIC	0.0058	5.15	8.44	4.43e-4
AIC + coll.	6.07e-05	5.04	43.90	0.1695
u-AIC + coll	0.0053	5.86	11.65	4.92e-04

Table 4.1: Simulation results of AIC and u-AIC without and with a random collision of random duration, averaged over 100 randomized reaching tasks.

Monotonic increase of control input

The second row of the plots in fig. 4.6 displays the control action of one joint. During a collision, the control input computed by the AIC keeps increasing due to the integral nature of the control law employed (see eq. (4.7)). After the collision is removed, the controller necessarily overshoots. In the u-AIC, one can saturate only the integral term and not the entire control law. On average, the AIC has four times the overshoot after a collision with a comparable settling time to the u-AIC.

4.7.2. 7-DOF Panda arm

On the real Panda arm, we compared 1) the reference tracking in configuration space, and 2) the collision behavior in terms of overshoot and commanded control

actions resulting from holding the robot arm away from its desired set point. The behaviors of the controllers are best appreciated in the accompanying video[1].

Reference tracking

The performance in terms of reference tracking of a sinusoidal wave is reported in fig. 4.7. The AIC never converges to the reference trajectory, and this is independent of the initial conditions. We highlight this by plotting the response of the two controllers after running them for 4 seconds.
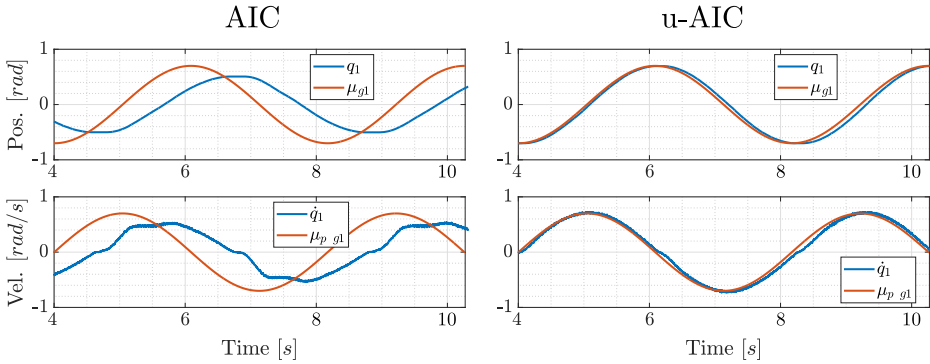
Figure 4.7: Experiments with real Panda arm on reference tracking. For readability, we only report the first joint.

The AIC performs poorly in terms of tracking errors. This is due to three main reasons: 1) the AIC [21, 109] only allows for position reference through its generative model, while the u-AIC can perform position and velocity tracking; 2) a more aggressive tuning of the AIC would result in high overshoots in case of collision, compromising safety.

Collision behavior

The collision behavior is reported in fig. 4.8. The robot is commanded to keep an initial position while a person is pushing, pulling, and holding the robot.

The AIC shows high overshoot which might be dangerous or damage delicate products such as fruits and vegetables in our setting. The u-AIC is instead well-behaved during interaction (see the attached video for a visualization of the results). The same behavior is observed when the robot is disturbed while performing a complicated trajectory.

4.8. Conclusion

In this paper, we discussed the fundamental limitations of the AIC for robot control and how the u-AIC overcomes them. These limitations arise from the fact that the state estimation is biased towards the goal and that the control action is not explicitly modeled in the generative model. These cause degraded state estimation

---

[1] https://www.youtube.com/watch?v=jI-zX8XvfgI
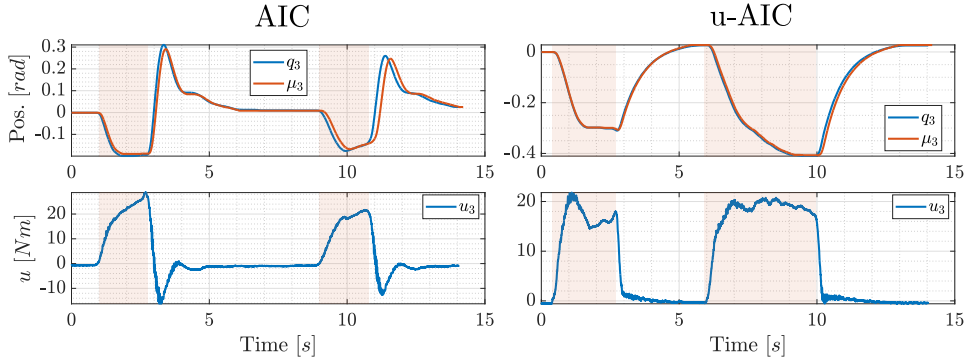
Figure 4.8: Experiments with real Panda arm during unwanted interaction (orange area). For readability we only report the third joint, being the most affected one. Faster convergence to zero steady-state error for the u-AIC can be achieved by tuning the integral action.

and can lead to large overshoots during human-robot interaction. We thoroughly discussed and extended the u-AIC providing a missing proof of convergence. We theoretically demonstrated the limitations of the AIC and provided experimental evidence of how the u-AIC overcomes them, both in simulation and the real world with a 7-DOF robot manipulator.

5

Active Inference and Behavior Trees for Reactive Action Planning and Execution in Robotics

*I*N *the previous two chapters, the focus was primarily on the adaptive and fault-tolerant aspects of Active Inference for low-level control. However, the resulting behaviors were constrained to point-to-point motions of the arm. In this chapter, we shift our attention toward utilizing the discrete formulation of Active Inference for high-level symbolic reasoning and action planning. We combine Active Inference and behavior trees to blend offline planning with online decision-making, obtaining a fast and reactive scheme for long-term tasks. Unlike conventional approaches that plan actions to execute, we employ behavior trees to plan desired states to be accomplished. This serves as a skeleton for guiding the online search for an action sequence using Active Inference. Significantly, this work pioneers the application of discrete Active Inference on a real robot.*

5.1. Abstract

We propose a hybrid combination of Active Inference and behavior trees (BTs) for reactive action planning and execution in dynamic environments, showing how robotic tasks can be formulated as a free-energy minimization problem. The proposed approach allows handling partially observable initial states and improves the robustness of classical BTs against unexpected contingencies while at the same time reducing the number of nodes in a tree. In this work, we specify the nominal behavior offline, through BTs. However, in contrast to previous approaches, we introduce a new type of leaf node to specify the desired state to be achieved rather than an action to execute. The decision of which action to execute to reach the desired state is performed online through Active Inference. This results in continual online planning and hierarchical deliberation. By doing so, an agent can follow a predefined offline plan while still being able to locally adapt and take autonomous decisions at runtime, respecting safety constraints. We provide proof of convergence and robustness analysis, and we validate our method in two different mobile manipulators performing similar tasks, both in a simulated and real retail environment. The results showed improved runtime adaptability with a fraction of the hand-coded nodes compared to classical BTs.

5.2. Introduction

Deliberation and reasoning capabilities for acting are crucial parts of online robot control, especially when operating in dynamic environments to complete long-term tasks. Over the years, researchers developed many task planners with various degrees of optimality, but little attention has been paid to actors [59, 95], i.e. algorithms endowed with reasoning and deliberation tools during plan execution. Previous literature [59, 95] advocates for a change in focus, explaining why this lack of actors could be one of the main causes of the limited spread of automated planning applications. Colledanchise and coauthors [32, 31, 117] proposed the use of BTs as graphical models for more reactive task execution, showing promising results. Other authors also tried to answer the call for actors [106, 57], but there are still open challenges to be addressed. These challenges have been identified and highlighted by many researchers, and can be summarized in two properties that an actor should possess [59, 31]:

- *Hierarchical deliberation*: each action in a plan may be a task that an actor may need to further refine online.

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

Actors should not be mere action executors then, but they should be capable of intelligently making decisions. This is particularly useful for challenging problems such as mobile manipulation in dynamic environments, where actions planned offline are prone to fail. In this paper, we consider mobile manipulation tasks in a retail environment with a partially observable initial state. We propose an actor based on

Active Inference. Such an actor is capable of following a task planned offline while still being able to make autonomous decisions at run-time to resolve unexpected situations.

Active Inference is a neuroscientific theory that has recently shown its potential in control engineering and robotics [88, 6, 107, 9], particularly in real-world experiments for low-level adaptive control [109, 98]. Active Inference describes a biologically plausible algorithm for perception, action, planning, and learning. This theory has been initially developed for continuous processes [48, 21, 19] where the main idea is that the brain's cognition and motor control functions could be described in terms of *free-energy minimization* [48]. In other words, we, as humans, take actions in order to fulfill prior expectations about a *desired prior* sensation [54]. Active Inference has also been extended to Markov decision processes for discrete decision making [51], recently gathering more and more interest [47, 118, 120, 72]. In this formulation, Active Inference is proposed as a unified framework to solve the exploitation-exploration dilemma by acting to minimize the free-energy. Agents can solve complicated problems once provided a *context sensitive prior* about preferences. Probabilistic beliefs about the state of the world are built through Bayesian inference, and a finite horizon plan is selected in order to maximize the evidence for a model that is biased toward the agent's preferences. At the time of writing, the use of discrete Active Inference for symbolic action planning is limited to low dimensional and simplified simulations [120, 72]. In addition, current solutions rely on fundamental assumptions such as instantaneous actions without preconditions, which do not hold in real robotic situations.

To tackle these limitations of Active Inference and to address the two main open challenges of hierarchical deliberation and continual planning, we propose a hybrid combination of Active Inference and BTs. We then apply this new idea to mobile manipulation in a dynamic retail environment.

5.2.1. Related Work

In this section, we mainly focus on related work on *reactive action planning and execution*, a class of methods that exploits reactive plans, which are stored structures that contain the behavior of an agent. To begin with, BTs [32, 96] gathered increasing popularity in robotics to specify reactive behaviors. BTs are de-facto replacing finite state machines (FSM) in several state-of-the-art systems, for instance in the successor of the ROS Navigation Stack, Nav2 [84]. BTs are graphical representations for action execution. The general advantage of BTs is that they are modular and can be composed into more complex higher-level behaviors, without the need to specify how different BTs relate to each other. They are also an intuitive representation that modularizes other architectures such as FSM and decision trees, with proven robustness and safety properties [32]. These advantages and the structure of BTs make them particularly suitable for the class of dynamic problems we are considering in this work, as explained later on in Section 5.3. However, in classical formulations of BTs, the plan reactivity still comes from hard-coded recovery behaviors. This means that highly reactive BTs are usually big and complex and that adding new robotic skills would require revising a large tree. To partially cope with these prob-

lems, Colledanchise and coauthors [31] proposed a blended reactive task and action planner that dynamically expands a BT at runtime through back-chaining. The solution can compensate for unanticipated scenarios, but cannot handle partially observable environments and uncertain action outcomes. Conflicts due to contingencies are handled by prioritizing (shifting) sub-trees. Safronov and coauthors [117] extended this and showed how to handle uncertainty in the BT formulation as well as planning with non-deterministic outcomes for actions and conditions. Other researchers combined the advantages of BTs with the theoretical guarantees on the performance of planning with the Planning Domain Definition Language (PDDL) by representing robot task plans as robust logical-dynamical systems (RLDS) [106]. RLDS results in a more concise problem description with respect to using BTs only, but online reactivity is again limited to the scenarios planned offline. For unforeseen contingencies, re-planning would be necessary, which is more resource-demanding than reacting [106]. The output of RLDS is equivalent to a BT, yet BTs remain more intuitive to compose and have more support from the community with open-source libraries and design tools.

Goal-Oriented Action Planning (GOAP) [67], instead, focuses on online action planning. This technique is used for non-player-characters (NPC) in video games [100]. Goals in GOAP do not contain predetermined plans. Instead, GOAP considers atomic behaviors which contain preconditions and effects. The general behavior of an agent can then be specified through very simple Finite State Machines (FSMs) because the transition logic is separated from the states themselves. GOAP generates a plan at run-time by searching in the space of available actions for a sequence that will bring the agent from the starting state to the goal state. However, GOAP requires hand-designed heuristics that are scenario-specific and it is computationally expensive for long-term tasks.

Hierarchical task Planning in the Now (HPN) [70] is an alternate plan and execution algorithm where a plan is generated backward starting from the desired goal, using A*. HPN recursively executes actions and re-plans. To cope with the stochasticity of the real world, HPN has been extended to belief HPN (BHPN) [71]. A follow-up work [80] focused on the reduction of the computational burden by implementing *selective re-planning* to repair local poor choices or exploit new opportunities without the need to re-compute the whole plan which is very costly since the search process is exponential in the length of the plan.

A different method for generating plans in a top-down manner is the Hierarchical Task Network (HTN) [38]. At each step, a high-level task is refined into lower-level tasks. In practice [58], the planner exploits a set of standard operating procedures for accomplishing a given task. The planner decomposes the given task by choosing among the available ones until the chosen primitive is directly applicable to the current state. Tasks are then iteratively replaced with new task networks. An important remark is that reactions to failures, time-outs, and external events are still a challenge for HTN planners. HTN requires the designer to write and debug potentially complex domain-specific recipes and, in very dynamic situations, re-planning might occur too often.

Finally, Active Inference is a normative principle underwriting perception, ac-

tion, planning, decision-making, and learning in biological or artificial agents. Active Inference on discrete state spaces [34] is a promising approach to solving the exploitation-exploration dilemma and empowers agents with continuous deliberation capabilities. The application of this theory, however, is still in an early stage for discrete decision-making where current works only focus on simplified simulations as proof of concept [47, 72, 118, 120]. Kaplan and Friston [72], for instance, simulated an artificial agent that had to learn and solve a maze given a set of simple possible actions to move (*up, down, left, right, stay*). Actions were assumed instantaneous and always executable. In general, current discrete Active Inference solutions lack a systematic and task-independent way of specifying prior preferences, which is fundamental to achieving a meaningful behavior, and they never consider action preconditions that are crucial in real-world robotics. As a consequence, plans with conflicting actions that might arise in dynamic environments are never addressed in the current state-of-the-art.

5.2.2. Contributions
In this work, we propose the hybrid combination of BTs and Active Inference to obtain more reactive actors with hierarchical deliberation and continual online planning capabilities. We introduce a method to include action preconditions and conflict resolution in Active Inference, as well as a systematic way of providing prior preferences through BTs. The proposed hybrid scheme leverages the advantages of online action selection with Active Inference and it removes the need for complex predefined fallback behaviors while providing convergence guarantees. In addition, we provide extensive mathematical derivations, examples, and code, to understand, test, and reproduce our findings.

5.2.3. Paper structure
The remainder of the paper is organized as follows. In Section 5.3 we provide an extensive background on Active Inference and BTs. Our novel hybrid approach is presented in Section 5.4, and its properties in terms of robustness and stability are analyzed in Section 5.5. In Section 5.6 we report the experimental evaluation, showing how our solution can be used with different mobile manipulators for different tasks in a retail environment. Finally, Section 5.7 contains the discussion, and Section 5.8 the conclusions.

5.3. Background on Active Inference and BTs
5.3.1. Background on Active Inference
Active Inference provides a unifying theory for perception, action, decision-making, and learning in biological or artificial agents [34]. Our Active Inference agent rests on the tuple $(\mathcal{O}, \mathcal{S}, \mathcal{A}, P, Q)$. This is composed of: a finite set of observations $\mathcal{O}$, a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a generative model P and an approximate posterior Q.

Active Inference proposes a solution for action and perception by assuming that actions will fulfill predictions that are based on inferred states of the world, given

some observations. The generative model contains beliefs about future states and action plans, where plans that lead to preferred observations are more likely. Perception and action are achieved through the optimization of two complementary objective functions, the variational free-energy F, and the expected free-energy G. These quantities to optimize are derived based on the generative model and the approximate posterior, as detailed later. Variational free-energy measures the fit between the generative model and past and current sensory observations, while expected free-energy scores future possible courses of action according to prior preferences and predicted observations. Fig. 5.1 depicts the general high-level idea. For a first-time reader of Active Inference, we advise consulting previous neuroscientific publications for a more extensive introduction [47].
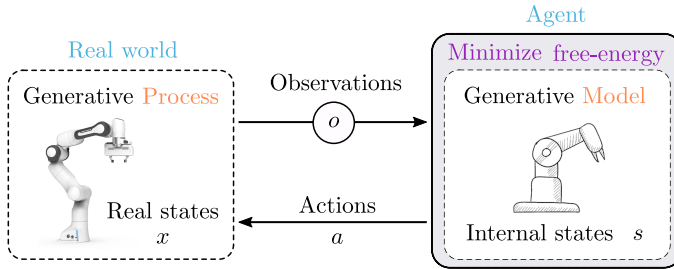
Figure 5.1: High-level visualization of an Active Inference agent. The generative process describes the true causes of the agent's observations o. An agent can apply actions to change the state of the world and to get observations that are aligned with its internal preferences.

In the following, we explain the form of the generative model and how the model parameters relate to states, actions, and observations. Based on this model, we present the expressions for the free-energy and expected free-energy that are used to derive the equations for perception and decision-making. We complement the theory with *pen-and-paper* examples and associated Python code[1]. All the other necessary mathematical derivations are added in the appendices.

Generative model

In Active Inference [47], the generative model P is chosen to be a Markov process that allows us to infer the states of the environment and to predict the effects of actions as well as future observations. This is expressed as a joint probability distribution $P(\bar{o}, \bar{s}, \boldsymbol{\eta}, \pi)$, where $\bar{o}$ is a sequence of observations, $\bar{s}$ is a sequence of states, $\boldsymbol{\eta}$ represents model parameters, and π is a plan. In particular, once given fixed model parameters $\boldsymbol{\eta}$ for a task[2], we can write:

$$P(\bar{o}, \bar{s}, \pi) = P(\pi) \prod_{\tau=1}^{T} P(s_\tau | s_{\tau-1}, \pi) P(o_\tau | s_\tau) \tag{5.1}$$

---

[1] https://github.com/cpezzato/discrete_active_inference/blob/main/discrete_ai/scripts/paper_examples.py

[2] Note that, in the general case, model parameters are not fixed and can be updated as well through Active Inference [47].

The full derivation of the generative model in eq. (5.1), the assumptions under its factorization, and how this joint probability is used to define the free-energy F can be found in Appendix A.1. The probability distributions in eq. (5.1) are represented internally by an Active Inference agent through the following parameters:

- $\boldsymbol{A} \in [0,1]^{r,m}$ is a matrix representation of the conditional probability $P(o_\tau|s_\tau)$, where r is the number of possible observations and m the number of possible states. $\boldsymbol{A}$ is also called the likelihood matrix, and it indicates the probability of observations given a specific state. Each column of $\boldsymbol{A}$ is a categorical distribution. It holds that $P(o_\tau|s_\tau, \boldsymbol{A}) = Cat(\boldsymbol{A}s_\tau)$. For a generic entry $\boldsymbol{A}_{ij} = P(o_\tau = i|s_\tau = j)$.

- $\boldsymbol{B} \in [0,1]^{m,m}$ represents a transition matrix. In particular $P(s_{\tau+1}|s_\tau, a_\tau) = Cat(\boldsymbol{B}_{a_\tau}s_\tau)$. For a symbolic action a_τ in a plan π, $\boldsymbol{B}_{a_\tau}$ represents the probability of state $s_{\tau+1}$ while applying action a_τ from state s_τ. The columns of $\boldsymbol{B}_{a_\tau}$ are categorical distributions.

- π is a sequence of actions over a time horizon T. $\boldsymbol{\pi}$ is the posterior distribution, a vector holding the probability of different plans. These probabilities depend on the expected free-energy in future time steps under plans given the current belief: $P(\pi) = \sigma(-G(\pi))$. Here, σ indicates the softmax function used to normalize probabilities.

An Active Inference agents contains also a model $\boldsymbol{D} \in [0,1]^m$ that represents the belief about the initial state at $\tau = 1$. So $P(s_0) = Cat(\boldsymbol{D})$. Additionally, an agent also represents prior preferences about desired observations for goal-directed behavior in $\boldsymbol{C} \in \mathbb{R}^r$, such that $P(o_\tau) = \boldsymbol{C}$.

In the context of this paper, we do not consider additional generative model parameters such as the $\boldsymbol{E}$ vector to encode priors over plans used to represent habits [126, 62]. The parameter $\boldsymbol{E}$ could be used to include common sense knowledge in the decision-making process, but this will be addressed in future work. Table 5.1 summarises the notation adopted in this paper. The top part contains quantities computed by Active Inference, while the bottom part the domain parameters required. As explained next, these internal models will be used by an Active Inference agent to compute the free-energy and the expected free-energy.

Variational Free-energy

Given the generative model as before, one can derive an expression for the variational free-energy. By minimizing F, an agent can determine the most likely hidden states given sensory information. The expression for F is given by:

$$F(\pi) = \sum_{\tau=1}^{T} \boldsymbol{s}_\tau^{\pi\top} \left[\ln \boldsymbol{s}_\tau^\pi - \ln \left(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi \right) - \ln \left(\boldsymbol{A}^\top \boldsymbol{o}_\tau \right) \right] \tag{5.2}$$

where $F(\pi)$ is a plan specific free-energy. The logarithm is considered element-wise. For the derivations please refer to Appendix A.2.

Table 5.1: Notation for Active Inference

Symbol	Description
$s_\tau \in \{0,1\}^m$	One-hot encoding of the hidden state at time τ, where m is the number of mutually exclusive states in the discrete state space.
$\boldsymbol{s}_\tau^\pi \in [0,1]^m$	Posterior distribution over the state under a plan π, where the elements sum up to one.
$o_\tau \in \{0,1\}^r$	Observation at time τ that can have r mutually exclusive possible values.
$\boldsymbol{o}_\tau^\pi \in [0,1]^r$	Posterior distribution of observations under a plan.
π	Plan specifying a sequence of symbolic actions $\pi = [a_\tau,\ a_{\tau+1}, ..., a_T]^\top$, where T is the time horizon.
$\boldsymbol{\pi} \in [0,1]^p$	Posterior distribution over plans, where p is the number of different plans.
$F(\pi) \in \mathbb{R}$	Plan specific variational free-energy.
$\boldsymbol{F}_\pi \in \mathbb{R}^p$	$\boldsymbol{F}_\pi = (F(\pi_1), F(\pi_2), ...)^\top$ is a column vector containing the free-energy for every plan.
$G(\pi, \tau) \in \mathbb{R}$	Expected free-energy for a plan at time τ.
$\boldsymbol{G}_\pi \in \mathbb{R}^p$	$\boldsymbol{G}_\pi = (G(\pi_1), G(\pi_2), ...)^\top$ is a column vector containing the expected free-energy for every plan.
$\boldsymbol{A} \in [0,1]^{r \times m}$	Likelihood matrix, mapping from hidden states to observations $P(o_\tau \vert s_\tau, \boldsymbol{A}) = Cat(\boldsymbol{A}s_\tau)$.
$\boldsymbol{B}_{a_\tau} \in [0,1]^{m \times m}$	Transition matrix, $P(s_{\tau+1} \vert s_\tau,\ a_\tau) = Cat(\boldsymbol{B}_{a_\tau} s_\tau)$.
$\boldsymbol{C} \in \mathbb{R}^r$	Prior preferences over observations $P(o_\tau) = \boldsymbol{C}$.
$\boldsymbol{D} \in [0,1]^m$	Prior over initial states $P(s_0) = Cat(\boldsymbol{D})$.
σ	Softmax function.

Perception

According to Active Inference, both perception and decision-making are based on the minimization of free-energy. In particular, for state estimation, we take partial derivatives of F with respect to the states and set the gradient to zero. The posterior distribution of the state, conditioned by a plan, is given by:

$$\boldsymbol{s}_{\tau=1}^\pi = \sigma(\ln \boldsymbol{D} + \ln(\boldsymbol{B}_{a_\tau}^\top \boldsymbol{s}_{\tau+1}^\pi) + \ln(\boldsymbol{A}^\top \boldsymbol{o}_\tau)) \tag{5.3a}$$

$$\boldsymbol{s}_{1<\tau<T}^\pi = \sigma(\ln(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi) + \ln(\boldsymbol{B}_{a_\tau}^\top \boldsymbol{s}_{\tau+1}^\pi) + \ln(\boldsymbol{A}^\top \boldsymbol{o}_\tau)) \tag{5.3b}$$

$$\boldsymbol{s}_{\tau=T}^\pi = \sigma(\ln(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi) + \ln(\boldsymbol{A}^\top \boldsymbol{o}_\tau)) \tag{5.3c}$$

where σ is the softmax function. The column of $\boldsymbol{B}_{a_\tau}^\top$ are normalized. For the complete derivation, please refer to Appendix A.3. Note that when $\tau = 1$ it holds $\ln(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi) = \ln \boldsymbol{D}$. We provide below an example of state update and highlight the effect of uncertain action outcomes in the state estimation process.

Example 1. <u>*State estimation:*</u> *An Active Inference agent lives in a simple world*

composed of one state which can have two possible values. The only action the agent can do is to stay still ($\boldsymbol{B}_{idle}$), so $\pi = a_\tau \forall \tau$ with $a_\tau = idle$. However, there are some chances that unwanted transitions between states can occur. The agent can only predict one step ahead ($T = 2$) and it receives an observation $\boldsymbol{o}_{\tau=1}$ at the start, that is related to the state through $\boldsymbol{A}$. The agent has no prior information about the initial state or future observations, thus $\boldsymbol{D}$ is uniform as well as the initial guess about the posterior distributions of the state. This is modeled as follows:

$$\boldsymbol{A} = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \ \boldsymbol{B}_{idle} = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}, \ \boldsymbol{D} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\boldsymbol{o}_{\tau=1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \boldsymbol{o}_{\tau=2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \boldsymbol{s}^\pi_{\tau=1} = \boldsymbol{s}^\pi_{\tau=2} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

The update of the posterior distribution for state estimation is, according to (5.3)*:*

$$
\begin{aligned}
\boldsymbol{s}^\pi_{\tau=1} &= \sigma \left(\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln \left(\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix} \begin{bmatrix} .5 \\ .5 \end{bmatrix} \right) \right. \\
&\quad + \left. \ln \left(\begin{bmatrix} .9 & .1 \\ .1 & .9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \right) = \begin{bmatrix} .9 \\ .1 \end{bmatrix} \\
\boldsymbol{s}^\pi_{\tau=2} &= \sigma \left(\ln \left(\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix} \begin{bmatrix} .9 \\ .1 \end{bmatrix} \right) + \ln \left(\begin{bmatrix} .9 & .1 \\ .1 & .9 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) \right) \\
&= \begin{bmatrix} .74 \\ .26 \end{bmatrix}
\end{aligned}
$$

As commonly done in Active Inference literature, a small number (for instance e^{-16} [126]) is added when computing the logarithms. This prevents numerical errors in the case of $\ln(0)$. Note that, if there would be no uncertainty on the actions the agent can take, i.e. $\boldsymbol{B}_{idle}$ is the identity in this case, then the estimated hidden state at $\tau = 2$ would be $\boldsymbol{s}^\pi_{\tau=2} = [0.9, 0.1]^\top$. The agent would then be more confident under this action.

Expected Free-energy

Active Inference unifies action selection and perception by assuming that actions fulfill predictions based on inferred states. Since the internal model can be biased toward preferred states or observations (*prior desires*), Active Inference induces actions that will bring the current beliefs toward the preferred states. An agent builds beliefs about future states which are then used to compute the expected free-energy. The latter is necessary to evaluate alternative plans. Plans that lead to preferred observations are more likely. Preferred observations are specified in the model parameter $\boldsymbol{C}$. This enables action to realize the next (proximal) *observation* predicted by the plan that leads to (distal) goals. The expected free-energy for a plan π at time τ is given by:

$$G(\pi, \tau) = \underbrace{\boldsymbol{o}^{\pi\top}_\tau \left[\ln \boldsymbol{o}^\pi_\tau - \ln \boldsymbol{C} \right]}_{Reward\ seeking} \underbrace{- diag(\boldsymbol{A}^\top \ln \boldsymbol{A})^\top \boldsymbol{s}^\pi_\tau}_{Information\ seeking} \tag{5.4}$$

The *diag()* function simply takes the diagonal elements of a matrix and puts them in a column vector. This is just a method to extract the correct matrix entries in order to compute the expected free-energy [126]. By minimizing expected free-energy the agent balances reward and information seeking (see Appendix A.4 for derivations). Reward and information-seeking behaviors that arise from the formulation of the expected free-energy are illustrated respectively in Examples 2 and 3. To keep the examples reasonably simple to be computed by hand, we consider that the posterior over states according to different plans have already been computed following (5.3), and are given.

Example 2. *Reward seeking: Consider an agent has computed the posterior distribution of the states $s_\tau^{\pi_1}$, $s_\tau^{\pi_2}$ under two different plans π_1 and π_2. The agent has a preference for a particular value of the state encoded in C. The model for this example is the following:*

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \; C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; s_\tau^{\pi_1} = \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix}, \; s_\tau^{\pi_2} = \begin{bmatrix} 0.05 \\ 0.95 \end{bmatrix}$$

Let us consider the reward-seeking term in (5.4) *(note that the information-seeking term is equal in both plans for this example). The observations expected under the two different plans are:*

$$o_\tau^{\pi_1} = A s_\tau^{\pi_1} = \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix}, \; o_\tau^{\pi_2} = A s_\tau^{\pi_2} = \begin{bmatrix} 0.14 \\ 0.86 \end{bmatrix}$$

Intuitively, according to C, the first plan is preferable and it should have the lowest expected free-energy because it leads to preferred observations with higher probability. Numerically:

$$o_\tau^{\pi_1 \top} [\ln o_\tau^{\pi_1} - \ln C] = \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix}^\top \left[\ln \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix} - \ln \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right] \approx 1.84$$

Similarly for π_2, $o_\tau^{\pi_2 \top} [\ln o_\tau^{\pi_2} - \ln C] \approx 13.35$. As can be noticed, the plan that brings the posterior state closest to the preference specified in C leads to the lowest reward-seeking term.

Example 3. *Information seeking: Let us consider a variation of Example 2. The agent is not given any preference for a specific state, and the likelihood matrix A encodes now the fact that observations are expected to provide more precise information when the agent is in the second state (second column of A). This results in the following models:*

$$A = \begin{bmatrix} 0.7 & 0.1 \\ 0.3 & 0.9 \end{bmatrix}, \; C = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; s_\tau^{\pi_1} = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, \; s_\tau^{\pi_2} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

We expect that the plan that leads to a state with less ambiguous information has the lowest information-seeking term. For the first plan:

$$- \; diag(A^\top \ln A)^\top s_\tau^{\pi_1} =$$

$$- \; diag\left(\begin{bmatrix} 0.7 & 0.3 \\ 0.1 & 0.9 \end{bmatrix} \ln \begin{bmatrix} 0.7 & 0.1 \\ 0.3 & 0.9 \end{bmatrix} \right)^\top \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} \approx 0.58$$

For the second plan, the information-seeking term is instead ≈ 0.35. The state achieved with the second plan generates less ambiguous observations. Plans that lead to the lowest ambiguity in sensory information, and thus minimize G, are preferred.

In more complex examples, minimizing G leads to a balance in reward and information seeking. For a fully-fledged exploration-exploitation problem, we refer the reader to a recently released Python library for Active Inference [61] which contains an interactive and visual example[3] of the emergent behavior of a rat in a grid world which has to collect cues to disclose the location of the reward.

Planning and decision making   Taking the gradient of F with respect to plans, and recalling that the generative model specifies the approximate posterior over plans as a softmax function of the expected free-energy [34] it holds that:

$$\boldsymbol{\pi} = \sigma(-\boldsymbol{G}_\pi - \boldsymbol{F}_\pi) \tag{5.5}$$

where the vector $\boldsymbol{\pi}$ encodes the posterior distribution over plans reflecting the predicted value of each plan. $\boldsymbol{F}_\pi = (F(\pi_1), F(\pi_2), ...)^\top$ and $\boldsymbol{G}_\pi = (G(\pi_1), G(\pi_2), ...)^\top$. See Appendix A.5 for the details.

Plan independent state-estimation   Given the probability over p possible plans, and the plan dependent states $\boldsymbol{s}_\tau^\pi$, we can compute the overall probability distribution for the states over time through the Bayesian Model Average:

$$\boldsymbol{s}_\tau = \sum_i \boldsymbol{s}_\tau^{\pi_i} \boldsymbol{\pi}_i, \ \text{where } i \in \{1, ..., p\} \tag{5.6}$$

were $\boldsymbol{s}_\tau^{\pi_i}$ is the probability of a state at time τ under plan i and $\boldsymbol{\pi}_i$ is the probability of plan i. This is the average prediction for the state at a certain time, so $\boldsymbol{s}_\tau$, according to the probability of each plan. In other words, this is a weighted average over different models. Models with high probability receive more weight, while models with lower probabilities are discounted.

Action selection   The action for the agent to be executed is the first action of the most likely plan:

$$\lambda = \max(\underbrace{[\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_p]}_{\boldsymbol{\pi}^\top}), \ a_\tau = \pi_\lambda(\tau = 1) \tag{5.7}$$

where λ is the index of the most likely plan.

Example 4. *__Plan and action selection:__ By computing the expected free-energy of Example 2 using (5.4) and including the information seeking term, we obtain* $\boldsymbol{G}_\pi = [G(\pi_1), G(\pi_2)]^\top \approx [2.16, 13.68]^\top$. *For the sake of this example, let us assume*

---

[3]https://heins_pymdp_2022-rtd.readthedocs.io/en/latest/notebooks/cue_chaining_demo.html

the free-energy is equal for both plans, that is $\boldsymbol{F}_\pi = [F(\pi_1), F(\pi_2)]^\top \approx [1.83, 1.83]^\top$. The posterior distribution over plans is then:

$$\boldsymbol{\pi} = \sigma \left(- \begin{bmatrix} 2.16 \\ 13.68 \end{bmatrix} - \begin{bmatrix} 1.83 \\ 1.83 \end{bmatrix} \right) \approx \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix}$$

As can be seen, the most likely plan is the first one, in accordance with the conclusions of Example 2. The action to be applied by the agent is the first action of π_1.

The Active Inference algorithm is summarised in pseudo-code in Algorithm 1.

Algorithm 1: Action selection with Active Inference

1: Set $\boldsymbol{C}$ ▷ Prior preferences
2: **for** $\tau = 1 : T$ **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 (5.2)
6:   Update posterior state $\boldsymbol{s}_\tau^\pi$ ▷ Equation (5.3)
7:   Compute G for each plan   ▷ Equation (5.4)
8:   Bayesian model averaging   ▷ Equation (5.6)
9:   Action selection   ▷ Equation (5.7)
10: **end for**
11: **Return** a ▷ Preferred action

Multiple sets of states and observations
The Active Inference model introduced in this section can also handle multiple sets of independent states and observations [126]. A famous example in the Active Inference literature covers a rat in a T-maze [47]. The rat is seeking a reward (cheese) but the location of the cheese is only known after receiving a cue. In this case, one set of states encodes the location of the rat, while another set encodes the initially unknown location of the cheese. Each independent set of states is called a *state factor*. In the same way, there can be multiple sets of observations coming from different sensors. Each set is a different *observation modality*. The literature contains step-by-step tutorials on Active Inference with fully worked-out toy examples including multiple factors and modalities [126, 62]. We provide explicit models for our robotic case with multiple state factors and observations in Sec. 5.4.1.

5.3.2. Background on BTs
We now describe the high-level concepts on the basis of BTs according to previous work [96, 32]. These concepts will be useful for understanding the novel hybrid scheme proposed in the next section. A BT is a directed tree composed of nodes and edges that can be seen as a graphical modeling language. It provides a structured representation for the execution of actions that are based on conditions and
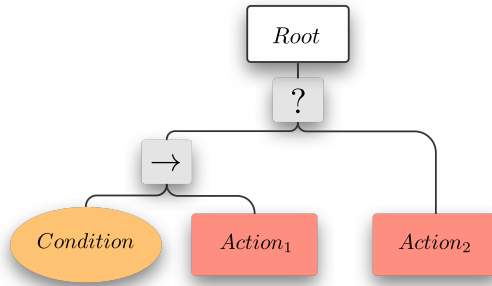
Figure 5.2: Example of BT. The *tick* traverses the tree starting from the *Root*. If *Condition* is true *Action*$_1$ is executed. Then, if *Action*$_1$ returns success, the *Root* returns success, otherwise *Action*$_2$ is executed.

observations in a system. The nodes in a BT follow the classical definition of parents and children. The root node is the only node without a parent, while the leaf nodes are all the nodes without children. In a BT, the nodes can be divided into control flow nodes (*Fallback*, *Sequence*, *Parallel*, or *Decorator*), and into execution nodes (*Action* or *Condition*) which are the leaf nodes of the tree. When executing a given BT in a control loop, the root node sends a *tick* to its child. A tick is nothing more than a signal that allows the execution of a child. The tick propagates in the tree following the rules dictated by each control node. A node returns a status to the parent, which can be *running* if its execution has not finished yet, *success* if the goal is achieved, or *failure* in the other cases. At this point, the return status is propagated back up the tree, which is traversed again following the same rules. The most important control nodes are:

Fallback nodes   A fallback node ticks its children from left to right. It returns *success* (or *running*) as soon as one of the children returns *success* (or *running*). When a child returns *success* or *running*, the fallback does not tick the next child, if present. If all the children return *failure*, the fallback returns *failure*. This node is graphically identified by a gray box with a question mark "?";

Sequence nodes   The sequence node ticks its children from left to right. It returns *running* (or *failure*) as soon as a child returns *running* (or *failure*). The sequence returns *success* only if all the children return *success*. If a child returns *running* or *failure*, the sequence does not tick the next child, if present. In the library, we used to implement our BTs [39] the *sequence* node, indicated with [→], keeps ticking a running child, and it restarts only if a child fails. Faconti and Colledanchise [39] provide also *reactive sequences* [→^{R}] where every time a sequence is ticked, the entire sequence is restarted from the first child.

The execution nodes are *Actions* and *Conditions*:

Action nodes An Action node performs an action in the environment. While an action is being executed, this node returns *running*. If the action is completed correctly it returns *success*, while if the action cannot be completed it returns *failure*. Actions are represented as red rectangles;

Condition nodes A Condition node determines if a condition is met or not, returning *success* or *failure* accordingly. Conditions never return *running* and do not change any states or variables. They are represented as orange ovals;

An example BT is given in Fig. 5.2.

5.4. Active Inference and BTs for Reactive Action Planning and Execution

In this section, we introduce our novel approach using BTs and Active Inference. Even though Active Inference is a very promising theory, from a computational perspective computing the expected free-energy for each possible plan that a robot might take is cost-prohibitive. This curse of dimensionality is due to the combinatorial explosion when looking deep into the future [34]. To solve this problem, we propose to replace *deep plans* with *shallow trees* that are *hierarchically composable*. This will allow us to simplify our offline plans, exploit opportunities, and act intelligently to resolve local unforeseen contingencies. Our idea consists of two main intuitions:

- To avoid combinatorial explosion while planning and acting with Active Inference for long-term tasks we specify the nominal behavior of an agent through a BT, used as a prior. In doing so, BTs provide *global reactivity to foreseen situations*

- To avoid coding every possible contingency in the BT, we program only desired states offline, and we leave action selection to the online Active Inference scheme. Active Inference provides then *local reactivity to unforeseen situations*.

To achieve such a hybrid integration, and to be able to deploy this architecture on real robotic platforms, we addressed the following three fundamental problems: 1) how to define the generative models for Active Inference in robotics, 2) how to use BTs to provide priors as desired states to Active Inference, 3) how to handle action preconditions in Active Inference and possible conflicts which might arise at run-time in a dynamic environment.

5.4.1. Definition of the models for Active Inference
The world in which a robot operates needs to be abstracted such that the Active Inference agent can perform its reasoning processes. In this work, we operate in a continuous environment with the ability to sense and act through symbolic decision-making. In the general case, the decision-making problem will include multiple sets of states, observations, and actions. Each independent set of states is a factor,

for a total of n_f factors. For a generic factor f_j where $j \in \mathcal{J} = \{1, ..., n_f\}$, the corresponding state factor is:

$$s^{(f_j)} = \left[s^{(f_j,1)}, s^{(f_j,2)}, ..., s^{(f_j,m^{(f_j)})}\right]^\top,$$
$$\mathcal{S} = \left\{s^{(f_j)} | j \in \mathcal{J}\right\} \tag{5.8}$$

where $m^{(f_j)}$ is the number of mutually exclusive symbolic values that a state factor can have. Each entry of $s^{(f_j)}$ is a real value between 0 and 1, and the sum of the entries is 1. This represents the current belief state. Then, we define $x \in \mathcal{X}$ as the continuous states of the world and the internal states of the robot are accessible through the symbolic perception system. The role of this perception system is to compute the symbolic observations based on the continuous state x, such that they can be manipulated by the discrete Active Inference agent. Observations o are used to build a probabilistic belief about the current state. Assuming one set of observations per state factor with $r^{(f_j)}$ possible values, it holds:

$$o^{(f_j)} = \left[o^{(f_j,1)}, o^{(f_j,2)}, ..., o^{(f_j,r^{(f_j)})}\right]^\top,$$
$$\mathcal{O} = \left\{o^{(f_j)} | j \in \mathcal{J}\right\} \tag{5.9}$$

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

$$a_\tau \in \alpha^{(f_j)} = \left\{a^{(f_j,1)}, a^{(f_j,2)}, ..., a^{(f_j,k^{(f_j)})}\right\},$$
$$\mathcal{A} = \left\{\alpha^{(f_j)} | j \in \mathcal{J}\right\} \tag{5.10}$$

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

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

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

Finally, we define the logical state $l^{(f_j)}$ as a one-hot encoding of $s^{(f_j)}$. We indicate as $\mathcal{L}_c(\tau) = \left\{l^{(f_j)} | j \in \mathcal{J}\right\}$ the (time varying) *current* logical state of the world. Defining a logic state based on the probabilistic belief s built with Active Inference, instead of directly using the observation of the states o, increases robustness against noisy sensor readings, as we will explain in Example 12. Given the model of the world just introduced, we can now define for each factor f_j the likelihood matrix $\boldsymbol{A}^{(f_j)}$, the $k^{(f_j)}$ transition matrices $\boldsymbol{B}_{a_\tau^{(f_j,\cdot)}}$ and the prior preferences $\boldsymbol{C}^{(f_j)}$. When an observation is available, $\boldsymbol{A}^{(f_j)}$ provides information about the corresponding value of a state factor $s^{(f_j)}$. For a particular state, the probability of a state observation pair

$o_\tau^{(f_j)}$, $s_\tau^{(f_j)}$ is given by $\boldsymbol{A}^{(f_j)} \in \mathbb{R}^{r^{(f_j)} \times m^{(f_j)}}$. In case a state factor is observable with full certainty, each state maps into the corresponding observation thus the likelihood matrix is the identity of size $m^{(f_j)}$, $I_{m^{(f_j)}}$. Note that knowing the mapping between observations and states does not necessarily mean that we can observe all the states at all times. Observations can be present or not, and when they are the likelihood matrix indicates the relation between that observation and the state. This relation can be more complex and incorporate uncertainty in the mapping as well. To define the transition matrices, we need to encode in a matrix form the effects of each action on the relevant state factors. The probability of a ending up in a state $s_{\tau+1}^{(f_j)}$, given $s_\tau^{(f_j)}$ and action $a_\tau^{(f_j,\cdot)}$ is given by:

$$P(s_{\tau+1}^{(f_j)}|s_\tau^{(f_j)}, a_\tau^{(f_j,\cdot)}) = Cat(\boldsymbol{B}_{a_\tau^{(f_j,\cdot)}} s_\tau^{(f_j)}),$$

$$\boldsymbol{B}_{a_\tau^{(f_j,\cdot)}} \in \mathbb{R}^{m^{(f_j)} \times m^{(f_j)}} \tag{5.11}$$

In other words, we define $\boldsymbol{B}_{a_\tau^{(f_j,\cdot)}}$ as a square matrix encoding the post-conditions of action $a_\tau^{(f_j,\cdot)}$. The prior preferences over observations (or states) need to be encoded, for each factor, in $\boldsymbol{C}^{(f_j)} \in \mathbb{R}^{m^{(f_j)}}$, with $\mathcal{C} = \{\boldsymbol{C}^{(f_j)} | j \in \mathcal{J}\}$. The higher the value of a preference, the more preferred a particular state is, and vice-versa. Priors are formed according to specific desires and they will be used to interface Active Inference and BTs. Finally, one also has to define the vector encoding the initial belief about the probability distribution of the states, that is $\boldsymbol{D}^{(f_j)} \in \mathbb{R}^{m^{(f_j)}}$. This vector is normalized, and when no prior information is available, each entry will be $1/m^{(f_j)}$. In this work, we assume the model parameters such as likelihood and transition matrices to be known. However, one could use the free-energy minimization to learn them [46].

Example 5. *Consider a mobile manipulator in a retail environment. We want the robot to be able to decide when to navigate to a certain goal location. To achieve so we need one state factor $s^{(loc)}$, one observation $o^{(loc)}$, and one symbolic action $a^{(loc,1)} = \texttt{moveTo(goal)}$. The robot can also decide not to do anything, so $a^{(loc,2)} = \texttt{idle}$.*

Actions	Preconditions	Postconditions
moveTo(goal)	-	$l^{(loc)} = [1\ 0]^\top$
idle	-	-

The current position in space of the robot is a continuous value $x \in \mathcal{X}$. However, during execution, the robot is given an observation $o^{(loc)}$ which indicates simply if the $\texttt{goal}$ has been reached or not. In this case $\boldsymbol{A}^{(loc)} = I_2$. The agent is then constantly building a probabilistic belief $s^{(loc)}$ encoding the chances of being at the $\texttt{goal}$. In case the robot has not yet reached the goal, a possible configuration at time τ is the following:

$$o^{(loc)} = \begin{bmatrix} \texttt{isAt(goal)} \\ \texttt{!isAt(goal)} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, s^{(loc)} = \begin{bmatrix} 0.08 \\ 0.92 \end{bmatrix}, \tag{5.12}$$

$$l^{(loc)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; \boldsymbol{B}_{moveTo} = \begin{bmatrix} 0.95 & 0.9 \\ 0.05 & 0.1 \end{bmatrix}, \; \boldsymbol{B}_{idle} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The preference over a state to be reached is given through $\boldsymbol{C}^{(loc)}$. A robot wanting to reach a location will have, for instance, a preference $\boldsymbol{C}^{(loc)} = [1,0]^\top$.

The transition matrix $\boldsymbol{B}_{moveTo}$ encodes the probability of reaching a goal location through the action `moveTo(goal)`, which might fail with a certain probability. We also encode an `Idle` action, which does not modify any state, but it provides information on the outcome of the action selection process as we will see in the next subsections. In this simple case, the world state is just a single state factor $s^{(loc)}$. On the other hand, in later more complicated examples, the world state will contain all the different aspects of the world, for which a probabilistic representation of their value is built and updated.

Using the proposed problem formulation for the Active Inference models, we can abstract redundant information that is unnecessary to make high-level decisions. For instance, in the example above, we are not interested in building a probabilistic belief of the current robot position. To decide if to use the action `moveTo(goal)` or not, it is sufficient to encode if the `goal` has been reached or not.

5.4.2. BTs integration: planning preferences, not actions

To achieve a meaningful behavior in the environment through Active Inference, we need to encode specific desires into the agent's brain through $\mathcal{C} = \left\{ \boldsymbol{C}^{(f_j)} | j \in \mathcal{J} \right\}$.

A prior as BT We propose to extend the available BT nodes in order to be able to specify desired states to be achieved as leaf nodes. We introduce a new type of leaf node called *prior nodes*, indicated with a green hexagon. These nodes can be seen as standard action nodes but instead of commanding an action, they simply set the desired value of a state in $\mathcal{C}$ and they run Active Inference for action selection. The prior node is then just a leaf node in the BT which returns: `Success` if a state is achieved, `Running` while trying to achieve it, or `Failure` if for some reason it is not possible to reach the desired state. The return statuses are according to the outcome of our reactive action selection process as explained in Section 5.4.4.

Sub-goals through BTs To reach a distal goal state, we plan achievable sub-goals in the form of desired logical states l, according to the available actions that a robot possesses. This idea of using sub-goals was already used in past work [72], but in our solution with BTs, we provide a task-independent way to define sub-goals which is only based on the set of available skills of the robot, such that we can make sure that it can complete the task. At planning time, we define the ideal sequence of states and actions to complete a task such that subsequent sub-goals (or logical desired states) are achievable by means of one action. This can be done manually or through automated planning. At run-time, however, we only provide the sequence of states to the algorithm, as in Fig. 5.3.
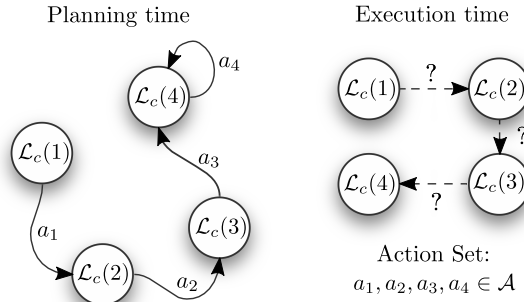
Figure 5.3: The path among states is planned offline using the available set of actions but only the sequence of states is provided at run-time. Actions are chosen online from the available set with Active Inference.

Example 6. *To program the behavior of the robot in Example 5 to visit a certain goal location, the BT will set the prior over $s^{(loc)}$ to $\boldsymbol{C}^{(loc)} = [1, 0]^\top$ meaning that the robot would like to sense to be at* `goal`.

A classical BT and a BT for Active Inference with prior nodes are reported in Fig. 5.4. Note that the action is left out in the BT for Active Inference because these are selected at runtime. In this particular case, the condition `isAt(goal)` can be seen as the desired observation to obtain.

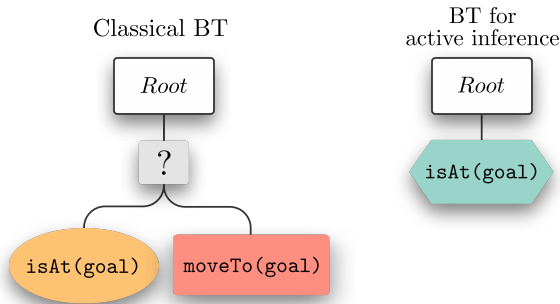Figure 5.4: BT to navigate to a location using a classical BT and a BT for Active Inference. One action `moveTo(goal)` is available and one condition `isAt(goal)` provides information if the current location is at the goal. The prior node for Active Inference (green hexagon) sets the desired prior and runs the action selection process.

Note that the amount of knowledge (i.e. number of states and actions) that is necessary to code a classical BT or our Active Inference version in Example 6 is the same. However, we abstract the fallback by planning in the state space and not in the action space. Instead of programming the action `moveTo(goal)` we only set a prior preference over the state `isAt(goal)` since the important information is retained in the state to be achieved rather than in the sequence of actions to
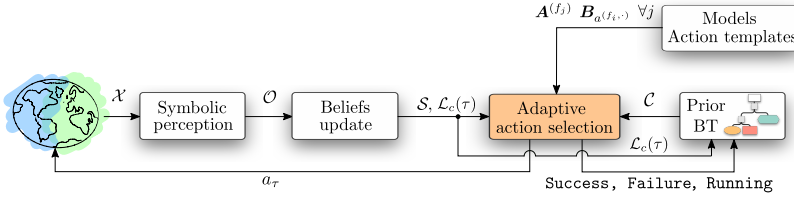
Figure 5.5: Overview of the control architecture for reactive action planning and execution using Active Inference. Adaptive action selection is performed according to Algorithm 2. The symbolic perception module computes the symbolic observations based on the continuous state. In this work, we assume this mapping is known and encode it through simple rules based on measurements from the robot's sensors. However, learning methods to define this relationship from data could be employed as well.

do so. Action selection through Active Inference will then select the appropriate skills to match the current state of the world with the desired one, minimizing this discrepancy through free-energy minimization.

Example 7. *Consider the scenario in Example 5 with the prior as in Example 6. The prior is specifying a preference over being at the desired goal location, but the mobile manipulator is not. The plan generated at runtime with Alg. 1 (see Examples 2 and 4) would be to perform the action* `moveTo(goal)`, *since this increases the probability of getting an observation* `isAt(goal)`.

As we will thoroughly explain in Sec. 5.5.1, our algorithm creates online a stable region of attraction from the current state to the goal state instead of planning it offline through fallbacks. See Example 11 for a concrete case.

5.4.3. Action preconditions and conflicts

Past work on Active Inference [72] was based on the assumption that actions were always executable and non-conflicting, but these do not hold in more realistic scenarios.

Action preconditions in Active Inference We propose to encode action preconditions as desired logical states that need to hold to be able to execute a particular action. This is illustrated in the next example.

Example 8. *We add one more action to the set of skills of our mobile manipulator:* `pick(obj)` *and the relative transition matrix* $\mathbf{B}_{pick}$. *The action templates are extended as follows:*

Actions	Preconditions	Postconditions
moveTo(goal)	–	$l^{(loc)} = [1\ 0]^\top$
		$l^{(reach)} = [1\ 0]^\top$
pick(obj)	isReachable(obj)	$l^{(hold)} = [1\ 0]^\top$

$$o^{(hold)} = \begin{bmatrix} isHolding(obj) \\ !isHolding(obj) \end{bmatrix}, \quad \boldsymbol{B}_{pick} = \begin{bmatrix} 0.95 & 0.9 \\ 0.05 & 0.1 \end{bmatrix}$$

$$o^{(reach)} = \begin{bmatrix} isReachable(obj) \\ !isReachable(obj) \end{bmatrix} \tag{5.13}$$

where we added a new logical state $l^{(hold)}$, the relative belief $s^{(hold)}$ and observation $o^{(hold)}$, which indicates if the robot is holding the object `obj`. In the simplest case, we suppose that the only precondition for successful grasping is that `obj` is reachable. We then add a logical state $l^{(reach)}$, as well as $s^{(reach)}$ and $o^{(reach)}$, to provide Active Inference with information about this precondition. $o^{(reach)}$ can be built for instance trying to compute a grasping pose for a given object. The robot can act on the state $l^{(hold)}$ through `pick`, and it can act on $l^{(reach)}$ through `moveTo`.

Conflicts resolution in Active Inference Conflict resolution due to dynamic changes in the environment and online decision-making is handled through modification of the prior preferences in $\mathcal{C}$. The BT designed offline specifies at runtime the desired state to be achieved. This is done by populating the prior preference over a state with the value of one. Note that if there is no goal, the preferences over states are set to zero everywhere so there is no incentive to act to achieve a different state. Given some preferences over states, the online decision-making algorithm with Active Inference selects an action, and checks if the preconditions are holding according to the current belief state. If so, the action is executed, if not, the missing preconditions are added to the current preferred state with a higher preference (i.e. > 1), in our case with value 2. This can lead to conflicts with the original BT [31], that is the robot might want to simultaneously achieve two conflicting states. However, the state relative to a missing precondition has higher priority (i.e. > 1) by construction. As explained in Algorithm 2, if preconditions are missing at runtime, action selection is performed again with the updated prior $\mathcal{C}$, such that actions that will satisfy them are more likely. With our method, there is no need to explicitly detect a conflict and re-ordering a BT then, as was done in past work on the dynamic expansion of BTs [31]. In fact, since the decision-making with Active Inference happens continuously during task execution, once a missing precondition is met this is removed from the current desired state. Thus, the only remaining preference is the one imposed by the BT, which can now be resumed. This leads to a natural conflict resolution and plan resuming without ad-hoc recovery mechanisms. The advantage of Active Inference is that we can represent *which* state is important but also *when* with different values of preference. Since missing preconditions are added to the current prior with a higher preference with respect to the offline plan, this will induce a behavior that can initially go against the initial BT because the new desire is more appealing to be satisfied. Conflict resolution is then achieved by *locally updating* prior desires about a state, giving them higher preference. The convergence analysis of this approach is reported in Section 5.5, and a concrete example of conflict resolution in a robotic scenario is presented in Sec. 5.6.4, Example 13.

5.4.4. Complete control scheme

Our solution is summarised in Algorithm 2 and Fig. 5.5. Every time a BT is ticked, given a certain frequency, Algorithm 2 is run. The symbolic perception layer takes the sensory readings and translates these continuous quantities into logical observations. This can be achieved through user-defined models according to the specific environment and sensors available. The logical observations are used to perform belief updating to keep a probabilistic representation of the world in $\mathcal{S}$. Then, the logical state $\mathcal{L}_c(\tau)$ is formed. Every time a *prior* node in the BT is ticked, the corresponding priors in $\mathcal{C}$ are set.

For both missing preconditions and conflicts, high-priority priors are removed from the preferences $\mathcal{C}$ whenever the preconditions are satisfied or the conflicts resolved (lines 6-10), allowing to resume the nominal flow of the BT. Active Inference from Algorithm 1 is then run for action selection. If no action is required since the logical state corresponds to the prior, the algorithm returns `Success`. Otherwise, the selected action's preconditions are checked and eventually pushed with higher priority. Then, action selection is performed with the updated prior. This procedure is repeated until either an executable action is found, returning `Running`, or no action can be executed, returning `Failure`. The case of `Failure` is handled through the global reactivity provided by the BT. This creates dynamic and stable regions of attraction as explained in Section 5.5.1, by means of sequential controller composition [23] (lines 17-31 of Algorithm 2).

Crucially, in this work, we propose the new idea of using *dynamic priors*. For a factor f_j, $\boldsymbol{C}^{(f_j)}$ is not fixed a priori as in past Active Inference works, but instead, it can change over time according to the BT for a task. This allows preconditions checking and conflict resolution within Active Inference. A robot can follow a long programmed routine while autonomously taking decisions to locally compensate for unexpected events. This considerably reduces the need for hard-coded fallbacks, allowing to compress the BT to a minimal number of nodes.

5.5. Theoretical analysis

5.5.1. Analysis of convergence

We provide a theoretical analysis of the proposed control architecture. There are two possible scenarios that might occur at run-time. Specifically, the environment might or might not differ from what has been planned offline through BTs. These two cases are analyzed in the following to study the convergence to the desired goal of our proposed solution.

The dynamic environment IS as planned

In a nominal execution, where the environment in which a robot is operating is the same as the one at planning time, there is a one-to-one equivalence between our approach and a classical BT formulation. This follows directly by the fact that the BT is defined according to Section 5.4.2, so each subsequent state is achievable by means of one single action. At any point of the task, the robot finds itself in the planned state and has only one preference over the next state given by the BT through $\mathcal{C}$. The only action which can minimize the expected free-energy is

---

Algorithm 2: Pseudo-code for Adaptive Action Selection

1: /* Get desired prior and parameters from BT */
2: $\mathcal{C}$, `param` $\leftarrow$ BT $\triangleright$ With priority 1
3: /* Set current observations, beliefs, and logical state */
4: Set $\mathcal{O}$, $\mathcal{S}$, $\mathcal{L}_c(\tau)$
5: /* Remove preferences with high-priority (i.e. > 1) if satisfied */
6: **for** all priors $\boldsymbol{C}^{(f_j)}$ with preference ≥ 1 **do**
7:    **if** $l_c^{(f_j)}$ holds **then**
8:       Remove pushed preference for $l_c^{(f_j)}$;
9:    **end if**
10: **end for**
11: /* Run Active Inference given $\mathcal{O}$, $\mathcal{S}$ and $\mathcal{C}$ */
12: $a_\tau \leftarrow$ `Action_selection`$(\mathcal{O}, \mathcal{S}, \mathcal{C})$ $\triangleright$ Alg. 1
13: Update $\mathcal{L}_c(\tau)$
14: **if** $a_\tau ==$ Idle **then**
15:    **return** Success; $\triangleright$ No action required
16: **else**
17:    /* Check action preconditions */
18:    **while** a_τ !=Idle **do**
19:       **if** $\text{prec}_{a_\tau} \in \mathcal{L}_c(\tau)$ OR $\text{prec}_{a_\tau} = \emptyset$ **then**
20:          Execute(a_τ);
21:          **break, return** Running; $\triangleright$ Executing a_τ
22:       **else**
23:          /* Push missing preconditions in $\mathcal{C}$ */
24:          $\mathcal{C} \leftarrow \text{prec}_{a_\tau}$; $\triangleright$ With priority 2
25:          /* Exclude a_τ and re-run Alg. 1 */
26:          Remove(a_τ);
27:          $a_\tau \leftarrow$ `Action_selection`$(\mathcal{O}, \mathcal{S}, \mathcal{C})$
28:          **if** $a_\tau ==$ Idle **then**
29:             **return** Failure; $\triangleright$ No solution
30:          **end if**
31:       **end if**
32:    **end while**
33: **end if**

---

the one used during offline planning. In a nominal case, then, we maintain all the properties of BTs, which are well explained in previous literature [32]. In particular, the behavior will be Finite-Time Successful (FTS) [32] if the atomic actions are assumed to return success after a finite time. Note that so far we did not consider actions with the same postconditions. However, in this case, Algorithm 2 would sequentially try all the alternatives following the given order at design time. This can be improved for instance by making use of semantic knowledge at runtime to inform the action selection process about preferences over actions to achieve the same outcome. This information can be stored for instance in a knowledge base and can be used to parametrize the generative model for Active Inference.

The dynamic environment IS NOT as planned

The most interesting case is when a subsequent desired state is not reachable as initially planned. As explained before, in such a case we push the missing preconditions of the selected action into the current prior $\mathcal{C}$ to locally and temporarily modify the goal. We analyze this idea in terms of sequential controllers (or actions) composition [23], and we show how Algorithm 2 generates a plan that will eventually converge to the initial goal. First of all, we provide some assumptions and definitions that will be useful for the analysis.

Assumption 1   The action templates with pre- and postconditions provided to the agent are correct;

Assumption 2   A given desired goal is achievable by at least one atomic action;

Definition 1   The domain of attraction of an action a_i is defined as the set of its preconditions. This domain for a_i is indicated as $\mathcal{D}(a_i)$;

Definition 2   We say that an action a_1 *prepares* action a_2 if the postconditions $\mathcal{P}_c$ of a_2 lie within the domain of attraction of a_1, so $\mathcal{P}_c(a_2) \subseteq \mathcal{D}(a_1)$;

Note   For the derivations in this section we consider, without lack of generality, one single factor such that we can drop the superscripts, i.e. $\boldsymbol{C}^{(f_j)} = \boldsymbol{C}$.

Following Algorithm 2 each time a prior leaf node is ticked in the BT, Active Inference is used to define a sequence of actions to bring the current state towards the goal. It is sufficient to show, then, that the asymptotically stable equilibrium of a generic generated sequence is the initial given goal.

Lemma 1. *Let $\mathcal{L}_c(\tau)$ be the current logic state of the world and $\mathcal{A}$ the set of available actions. An action $a_i \in \mathcal{A}$ can only be executed within its domain of attraction, so when $\mathcal{L}_c(\tau) \in \mathcal{D}(a_i)$. Let us assume that the goal encoded in $\boldsymbol{C}$ is a postcondition of an action a_1 such that $\mathcal{P}_c(a_1) = \boldsymbol{C}$, and that $\mathcal{L}_c(\tau) \neq \boldsymbol{C}$. If $\mathcal{L}_c(\tau) \notin \mathcal{D}(a_1)$, Algorithm 2 generates a plan $\pi = \{a_1, \dots, a_N\}$ with domain of attraction $\mathcal{D}(\pi)$ according to the steps below.*

1. *Let the initial sequence contain $a_1 \in \mathcal{A}$, $\pi(1) = \{a_1\}$, $\mathcal{D}(\pi) = \mathcal{D}(a_1)$, set $N = 1$*

2. *Remove a_N from the available actions, and add the unmet preconditions $\mathcal{D}(a_N)$ to the prior $\boldsymbol{C}$ with higher priority, such that $\boldsymbol{C} = \boldsymbol{C} \cup \mathcal{D}(a_N)$*

3. *Select a_{N+1} through Active Inference (Algorithm 1). Then, a_{N+1} prepares a_N by construction, $\pi(N+1) = \pi(N) \cup \{a_N\}$, $\mathcal{D}_{N+1}(\pi) = \mathcal{D}_N(\pi) \cup \mathcal{D}(a_{N+1})$, and $N = N + 1$*

4. *Repeat 2, 3 until $\mathcal{L}_c(\tau) \in \mathcal{D}(a_N)$ `OR` a_N == `Idle`*

If $\mathcal{L}_c(\tau) \in \mathcal{D}(a_N)$, the sequential composition π with region of attraction $\mathcal{D}(\pi) = \bigcup_{a_i} \mathcal{D}(a_i)$ for $i = 1, ..., N$ stabilizes the system at the given desired state $\boldsymbol{C}$. If $a_i \in \mathcal{A}$ are FTS, then π is FTS.

Proof. Since $\mathcal{L}_c(\tau) \in \mathcal{D}(a_N)$ and $\mathcal{P}_c(a_N) \subseteq \mathcal{D}(a_{N-1})$, it follows that $\mathcal{L}_c(\tau)$ is moving towards $\boldsymbol{C}$. Moreover, by construction $\mathcal{D}(a_1) \subseteq \bigcup_{a_i} \mathcal{P}_c(a_i)$ for $i = 2, ..., N$. After completing action a_1, it results $\mathcal{L}_c(\tau) \equiv \boldsymbol{C}$ since by definition $\mathcal{P}_c(a_1) = \boldsymbol{C}$. ∎

Note that if $\mathcal{L}_c(\tau) \in \mathcal{D}(a_N)$ does not hold after sampling all available actions, it means that the algorithm is unable to find a set of actions that can satisfy the preconditions of the initially planned action. This situation is a major failure that needs to be handled by the overall BT. *Lemma* 1 is a direct consequence of the sequential behavior composition of FTS actions where each action has effects within the domain of attraction of the action below. The asymptotically stable equilibrium of each controller is either the goal $\boldsymbol{C}$, or it is within the region of attraction of another action earlier in the sequence [32, 23, 93]. One can visualize the idea of sequential composition in Fig. 5.6.
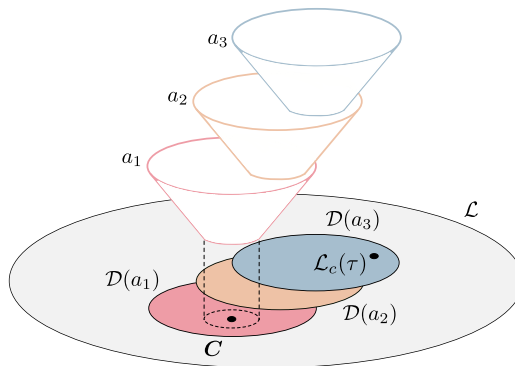

Figure 5.6: Schematic visualization of the domain of attraction $\mathcal{D}(\cdot)$ of different controllers around the current logical state $\mathcal{L}_c(\tau)$, as well as their postconditions within the domain of attraction of the controller below.

5.5.2. Analysis of robustness

It is not easy to find a common and objective definition of robustness to analyze the characteristics of algorithms for task execution. One possible way is to describe robustness in terms of domains or regions of attraction [23, 32]. When considering task planning and execution with classical BTs, often these regions of attraction are defined offline leading to a complex and extensive analysis of the possible contingencies that might eventually happen [32], and these are specific to each different task. Alternatively, adapting the region of attraction requires either re-planning [106] or dynamic BT expansion [31]. Robustness can be measured according to the size of this region, such that a robot can achieve the desired goal from a plurality of initial conditions. With Algorithm 2 we achieve robust behavior by *dynamically* generating a suitable region of attraction according to the minimization of free-energy. This region brings the current state towards the desired goal. We then cover only the necessary region in order to be able to steer the current state to the desired goal, changing prior preferences at run-time.

Corollary 1. *When an executable action a_N is found during task execution through Algorithm 2 such that $\pi = \{a_1, \ldots, a_N\}$, the plan has a domain of attraction towards a given goal that includes the current state $\mathcal{L}_c(\tau)$. If $\mathcal{D}(a_1) \subseteq \bigcup_{a_i} \mathcal{P}_c(a_i)$ for $i = 2, \ldots, N$ the plan is asymptotically stable.*

Proof. The corollary follows simply from *Lemma* 1. ■

Example 9. *Let us assume that Algorithm 2 produced a plan $\pi = \{a_1, a_2\}$, a set of FTS actions, where a_2 is executable so $\mathcal{L}_c(\tau) \in \mathcal{D}(a_2)$, and its effects are such that $\mathcal{P}_c(a_2) \equiv \mathcal{D}(a_1)$. Since a_2 is FTS, after a certain running time $\mathcal{L}_c(\tau) \in \mathcal{P}_c(a_2)$. The next tick after the effects of a_2 took place, $\pi = \{a_1\}$ where this time a_1 is executable since $\mathcal{L}_c(\tau) \in \mathcal{D}(a_1)$ and $\mathcal{P}_c(a_1) = \mathbf{C}$. The overall goal is then achieved in a finite time.*

Instead of looking for globally asymptotically stable plans from each initial state to each possible goal, which can be unfeasible or at least very hard [23], we define smaller regions of attractions dynamically, according to the current state and goal.

5.6. Experimental evaluation

In this section, we evaluate our algorithm in terms of robustness, safety, and conflict resolution in two validation scenarios with two different mobile manipulators and tasks. We also provide a theoretical comparison with classical and dynamically expanded BTs.

5.6.1. Experimental scenarios

Scenario 1

The task is to pick one object from a crate and place it on top of a table. This object might or might not be reachable from the initial robot configuration, and the placing location might or might not be occupied by another movable box. Crucially, the state of the table cannot be observed until the table is reached. This results in a partially

observable initial state at the start of the mission where the place location has a 50% chance of being either free or occupied. Additionally, we suppose that other external events, or agents, can interfere with the execution of the task, resulting in either helping or adversarial behavior. The robot used for the first validation scenario is a mobile manipulator consisting of a Clearpath Boxer mobile base, in combination with a Franka Emika Panda arm. The experiment for this scenario was conducted in a Gazebo simulation in a simplified version of a real retail store, see Fig. 5.7.
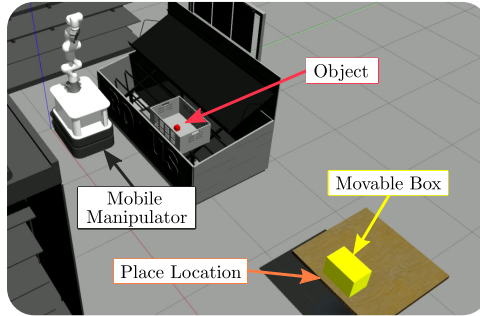
Figure 5.7: Simulation of the mobile manipulation task.

Scenario 2

The task is to fetch a product in a mockup retail store and stock it on a shelf using the real mobile manipulator TIAGo, as in Fig. 5.8.

Importantly, the BT for completing the task in the real store with TIAGo is the same one used for simulation with the Panda arm and the mobile base, just parametrized with a different object and place location. The code developed for the experiments and theoretical examples is publicly available[4].
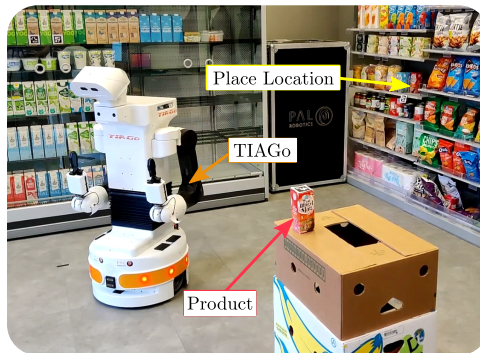

Figure 5.8: Experiments with TIAGo, stocking a product on the shelf.

---

[4]https://github.com/cpezzato/discrete_active_inference

5.6.2. Implementation
Models for Scenarios 1 and 2

In order to program the tasks for *Scenarios 1* and *2*, we extended the robot skills defined in our theoretical Example 8. We added then two extra states and their relative observations: `isPlacedAt(loc, obj)` called $s^{(place)}$, and `isLocationFree(loc)` called $s^{(free)}$. The state $s^{(place)}$ indicates whether or not `obj` is at `loc`, with associated probability, while $s^{(free)}$ indicates whether `loc` is occupied by another object. Then, we also had to add three more actions, which are 1) `place(obj, loc)`, 2) `push(obj)` to free a placing location, and 3) `placeOnPlate(obj)`, to place the object held by the gripper on the robot's plate. We summarise states and skills for the mobile manipulator in Table 5.2.

Table 5.2: Notation for states and actions

State, Boolean State	Description
$s^{(loc)}$, $l^{(loc)}$	Belief about being at the goal location
$s^{(hold)}$, $l^{(hold)}$	Belief about holding an object
$s^{(reach)}$, $l^{(reach)}$	Belief about reachability of an object
$s^{(place)}$, $l^{(place)}$	Belief about an object being placed at a location
$s^{(free)}$, $l^{(free)}$	Belief about a location being free

Actions	Preconditions	Postconditions
`moveTo(goal)`	–	$l^{(loc)} = [1\ 0]^\top$
`pick(obj)`	`isReachable(obj)` `!isHolding`	$l^{(reach)} = [1\ 0]^\top$ $l^{(hold)} = [1\ 0]^\top$
`place(obj,loc)`	`isLocationFree(loc)`	$l^{(place)} = [1\ 0]^\top$
`push()`	`!isHolding`	$l^{(free)} = [1\ 0]^\top$
`placeOnPlate()`	–	$l^{(hold)} = [0\ 1]^\top$

The likelihood matrices are just the identity, while the transition matrices simply map the postconditions of actions, similarly to Example 8. Note that the design of actions and states is not unique, and other combinations are possible. One can make atomic actions increasingly more complex, or add more preconditions. The plan, specified in a BT, contains the desired sequence of states to complete the task, leaving out from the offline planning other complex fallbacks to cope with contingencies associated with the dynamic nature of the environment. The BT for performing the tasks in both *Scenario 1* and *Scenario 2* is reported in Fig. 5.9. Note that the fallback for the action `moveTo` could be substituted by another prior node as in Fig. 5.4. However, we chose this alternative solution to highlight the hybrid combination of classical BTs and Active Inference. Design principles to choose when to use prior nodes and when normal fallbacks are reported in Sec.5.6.6.
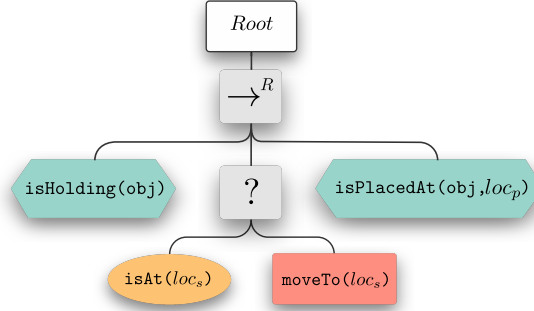
Figure 5.9: BT with prior nodes to complete the mobile manipulation task in the retail store, *Scenario 1* and 2. loc_s, loc_p are respectively the location in front of the shelf in the store and the desired place location of an item

Execution of Algorithm 2

We provide a full execution of Algorithm 2 in Example 10. We consider *Scenario 1,* for which the initial configuration of the robot is depicted in Fig. 5.7, and the BT for the task is the one if Fig. 5.9.

Example 10. *Let us consider Algorithm 2 and Scenario 1. At the start of the task, the first node `isHolding(obj)` in the BT is ticked, and the corresponding prior preference is set, $C^{(hold)} = [1,0]^\top$ (line 2 Alg. 2). Since the robot is not holding the desired `obj` and it is not reachable, $o^{(hold)} = o^{(reach)} = [0,1]^\top$. At the start, the initial states are a uniform distribution $s^{(hold)} = s^{(reach)} = [0.5, 0.5]^\top$ (line 4). Since the task just started, the only prior preference is the one set by the BT, so there are no high-priority priors (lines 6-10). Algorithm 1 is then run (line 12), updating the states $s^{(hold)}$, $s^{(reach)}$ according to the given observations, and selecting an action a_τ. In this example, the updated most probable logical state (line 13) will be $l_c^{(hold)} = l_c^{(reach)} = [0,1]^\top$ and a_τ will be `pick(obj)` since there is a mismatch between the $C^{(hold)}$ and $l_c^{(hold)}$. The preconditions of `pick(obj)` are checked (line 19). This action requires the object to be reachable, so this missing precondition is added to the preferences with high priority, that is $C^{(reach)} = [2,0]^\top$. Active Inference is run again with the updated prior (line 27). This process (lines 17-32) is repeated until either an executable action is found (lines 19-21) or the selected action is `idle` (lines 28-30). In the first case, a_τ is executed and the algorithm returns running. In the second case, a failure is returned indicating that no action can be performed to satisfy prior preferences. In this example, re-running Active Inference (line 27) with $C^{(hold)} = [1,0]^\top$ and $C^{(reach)} = [2,0]^\top$ would return the action `moveTo`. There are no preconditions for this action, thus it can be executed (lines 20-21). The BT keeps being ticked at a certain frequency, and until the object becomes reachable, the same steps as just described will be repeated. As soon as the robot can reach the object, so $l_c^{(reach)} = [1,0]^\top$, the preference for $C^{(reach)}$ is removed (lines 6-10) and set to $[0,0]^\top$. This time, when the action `pick` is selected, its preconditions are satisfied and thus it can be executed. After holding the*

object, when the BT is ticked no action is needed since the prior is already satisfied. Algorithm 2 returns success (lines 14-16) and the task can proceed with the next node in the BT.

Note that the BT designed for *Scenario 1* was entirely reused in *Scenario 2*, with the only adaptation of the desired object and locations in the BT. In Sec. 5.6.3 and Sec. 5.6.4, robustness and run-time conflicts resolution are analyzed for *Scenario 1*, but similar considerations can be derived for *scenario 2*.

5.6.3. Robustness: Dynamic regions of attraction

With our approach, we improve robustness compared to classical BTs in two different ways: 1) in terms of task execution, and 2) against noisy sensory readings. We elaborate on the following:

Robustness of task execution

With our algorithm we can generate complex regions of attraction dynamically at runtime, alleviating the burden of programming every fallback beforehand in a BT, which is prone to fail in edge cases that haven't been considered at design time. We illustrate this in the example below. Consider Example 10. According to the current world's state, Algorithm 2 selects different actions to generate a suitable domain of attraction:

Example 11. *The initial conditions are such that the object is unreachable. Let $s^{(hold)}$ be the probabilistic belief of holding an object, and $s^{(reach)}$ be the probabilistic belief of its reachability. The domain of attraction generated by Algorithm 2 at runtime is depicted in Fig. 5.10 using phase portraits [32]. Actions, when performed, increase the probability of their postconditions.*
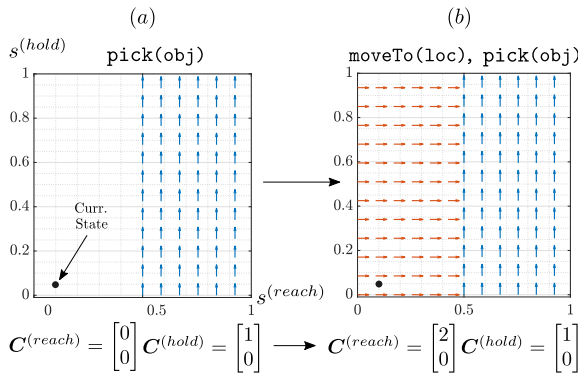

Figure 5.10: Dynamic domain of attraction generated by Algorithm 2 for Example 11. (a) relates to the action `pick(obj)`, and (b) is the composition of `moveTo(loc)` and `pick(obj)` after automatically updating the prior preferences

From Fig. 5.10, we can see that the goal of the Active Inference agent is to hold `obj` so $C^{(hold)} = [1\ 0]^\top$. The first selected action is then `pick(obj)`. However, since

the current logical state is not contained in the domain of attraction of the action, the prior preferences are updated with the missing (higher priority) precondition according to the action template provided, that is `isReachable` so $C^{(reach)} = [2, 0]^\top$. This results in a sequential composition of controllers with a stable equilibrium corresponding to the postconditions of `pick(obj)`. On the other hand, to achieve the same domain of attraction with a classical BT, one would require several additional nodes, as explained in Sec. 5.6.6 and visualized in Fig. 5.14. Instead of extensively programming fallback behaviors, Algorithm 2 endows our actor with deliberation capabilities and allows the agent to reason about the current state of the environment, the desired state, and the available action templates.

Robustness against noisy sensory readings

BTs are purely reactive which means that every action, or sub-behavior, is executed in response to an event or a condition determined at the current time step τ. If an instantaneous observation is erroneous, wrong transitions could be triggered because in classical BTs there is no notion and representation of a "state" that is maintained and updated over time. This might be a problem in the presence of noise in the observations.

Example 12. *Consider a generic fallback based on a condition check in a BT, as in Fig. 5.11. The perception system, on which the condition check is based produced at time τ an erroneous observation, for instance, due to poor lighting conditions in an object detection algorithm. If the condition is checked at time τ, a purely reactive system would produce a wrong transition because the condition returns failure, and the fallback would tick the action.*
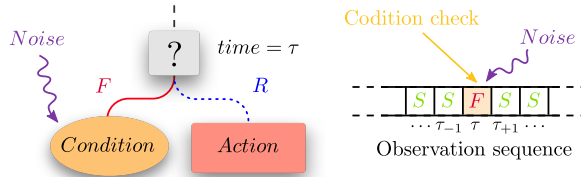

Figure 5.11: Erroneous transition in a purely reactive BT due to a noisy observation at time τ. F, S, and R mean respectively, *Failure, Success* and *Running*.

In Active Inference, the probabilistic representation of a state helps filter out this kind of spurious sensory input. For instance, a wrong observation like the one above would have caused the robot to be just slightly less confident about that state, with no erroneous transition.

5.6.4. Resolving run-time conflicts

Unexpected events affecting the system during action selection can lead to conflicts with the initial offline plan. This is the case in one execution of the mobile manipulation task as in Fig. 5.12 where after picking the object and moving in front of the table, the robot senses that the place location is not free. In this situation, a conflict

with the offline plan arises, where there is a preference for two mutually exclusive states, namely holding the red cube but also having the gripper free in order to empty the place location. We describe this situation more formally in Example 13, and we then explain how such a conflict is resolved.

Example 13. *For solving the situation in Fig. 5.12 using the BT in Fig. 5.9, the robot should 1) hold the object, 2) be at the desired place location, and 3) have the object placed. The preferences are planned offline and the BT populates the relative priors with a unitary preference at runtime (eq. (5.14a)). At this point of the execution, there is a mismatch between the current logical belief about the state $l^{(place)}$ and the desired one, in fact $C^{(place)}$ differs from $l^{(place)}$ (eq. (5.14b)) because the object is not placed at the desired location.*

$$C^{(hold)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ C^{(place)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{5.14a}$$

$$l^{(hold)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ l^{(place)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.14b}$$

The selected action with Active Inference in this situation is `place(obj,loc)`. The missing precondition on the place location to be free is added to the prior (see $C^{(free)}$ in eq. (5.15)) and the action selection is performed again. The only action that can minimize free-energy further is now `push`. Then, the missing precondition for this action (i.e. `!isHolding`) is added in the current prior with higher priority in $C^{(hold)}$ (line 24 in Algorithm 2).

$$C^{(free)} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \ C^{(hold)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{5.15}$$

The required `push` action to proceed with the task has a conflicting precondition with the offline plan, see $C^{(hold)}$ in eq. (5.15).
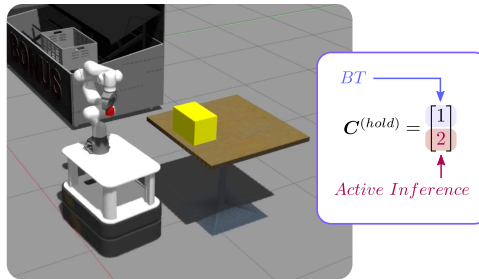

Figure 5.12: Example of conflict during mobile manipulation using the BT from Fig. 5.9. The BT is defining a preference of 1 over holding the red cube, but the runtime situation requires a free gripper to safely push away an unexpected object. Active Inference is then required to achieve an unmet precondition with higher preference.

Even though the desired state specified in the BT is `isHolding(obj)`, at this particular moment there is a higher preference for having the gripper free due to a

missing precondition to proceed with the plan. Algorithm 2 selects then the action that best matches the current prior desires, or equivalently that minimizes expected free-energy the most, that is `placeOnPlate` to obtain $l^{(hold)} = [0, \ 1]^\top$. This allows, then, to perform the action `push`. Once the place location is free after pushing, the high-priority preference on having the location free is removed from the prior. As a consequence, there are also no more preferences pushed with high priority over the state $l^{(hold)}$ which is only set by the BT as $\boldsymbol{C}^{(hold)} = [1, \ 0]^\top$. The red cube is then picked again and placed on the table since no more conflicts are present.

Videos of the simulations and experiments can be found online[5]. The BTs to encode priors for Active Inference are implemented using a standard library [39].

5.6.5. Safety

When designing adaptive behaviors for autonomous robots, attention should be paid to *safety*. The proposed algorithm allows to retain control over the general behavior of the robot and to force a specific routine in case something goes wrong leveraging the structure of BTs. In fact, we are able to include adaptation through Active Inference only in the parts of the task that require it, keeping all the properties of BTs intact. Safety guarantees for the whole task can easily be added by using a sequence node where the leftmost part is the safety criteria to be satisfied [32] by the right part of the behavior, as shown in Fig. 5.13. In this example, the BT
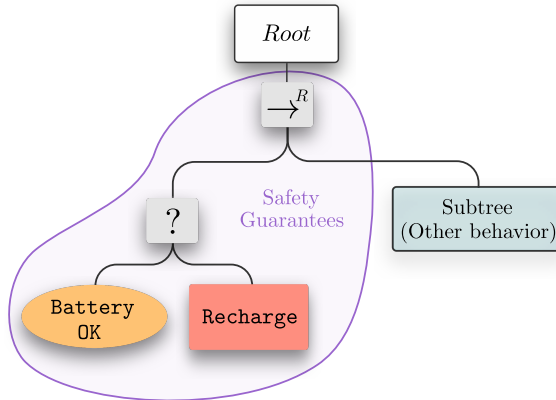

Figure 5.13: BT with safety guarantees while allowing runtime adaptation.

allows avoiding battery drops below a safety-critical value while performing a task. The sub-tree on the right can be any other BT, for instance, the one used to solve *Scenario 1* and *Scenario 2* from Fig. 5.9.

Since, by construction, a BT is executed from left to right, one can ensure that the robot is guaranteed to satisfy the leftmost condition first before proceeding with the rest of the behavior. In our specific case, this allows us to easily override the online decision-making process with Active Inference where needed, in favor of safety
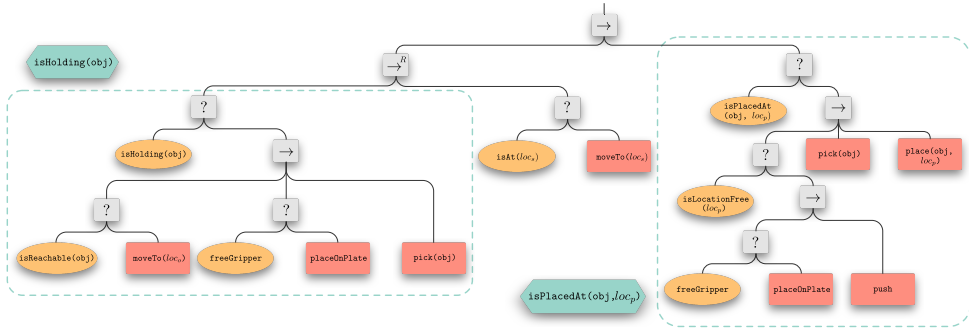
---

[5]https://youtu.be/dEjXu-sD1SI

Figure 5.14: Possible standard BT to perform *Scenario 1* and *Scenario 2* without prior nodes for Active Inference. Parts of the behavior that require several fallbacks can be substituted by prior nodes for online adaptation instead.

routines. Note that safety guarantees can also be provided in specific parts of the three only, and not necessarily for the whole tree. For example in Fig. 5.9, one might ensure navigation at a low speed only while transporting an object to a place location.

5.6.6. Comparison and design principles
Comparison with other BT approaches
The hybrid scheme of Active Inference and BTs aims to provide a framework for reactive action planning and execution in robotic systems. For this reason, we compare the properties of our approach with standard BTs [32] and with BTs generated through expansion from goal conditions [31]. *Scenario 1* and *Scenario 2* can be tackled, for instance, by explicitly planning every fallback behavior with classical BTs, as in Fig. 5.14. Even if this provides the same reactive behavior as the one generated by Fig. 5.9, far more (planning) effort is needed: to solve the same task one would require 12 control nodes, 8 condition nodes, and 7 actions, for a total of 27 nodes compared to the 6 needed in our approach that is an ∼ 88% compression.

Importantly, the development effort of a prior node in a BT is the same as a standard action node. It is true that Active Inference requires specifying the likelihood and transition matrices encoding actions pre- and postconditions, but this has to be done only once while defining the available skills of a robot, and it is independent of the task to be solved. Thus, a designer is not concerned with this when adding a prior node in a BT.

Instead of planning several fallbacks offline, BTs can dynamically be expanded from a single goal condition, through back-chaining, to blend planning and acting online [31]. To solve *Scenario 1* and *Scenario 2* with this approach, one needs to define a goal condition `isPlacedAt(obj, loc)` similarly to our solution, and define the preconditions of the action `place(obj, loc)` such that they contain the fact that the robot is holding the object, that the place location is reachable, and it is free. Then, to solve *Scenarios 1* and *2* one needs to define only the final goal condition and run the algorithm proposed by Colledanchise and coauthors [31]. Even

though this allows to complete tasks similar to what we propose, this strategy [31] comes with a fundamental theoretical limitation: adaptation cannot be selectively added only to specific parts of the tree. The whole behavior is indeed determined at runtime based on preconditions and effects of actions starting from a goal condition. The addition of safety guarantees can only happen for the whole task, and not in selected parts of the tree derived online.

To conclude, the hybrid combination of Active Inference and BTs allows for combining the advantages of both offline design of BTs and online dynamic expansion. In particular: it drastically reduces the number of necessary nodes planned offline in a BT, it can handle partial observability of the initial state, and it allows to selectively add adaptation and safety guarantees in specific parts of the tree.

Another important difference between our approach and other BTs solutions is that we introduced the concept of *state* in a BT through the *prior* node for which a probabilistic belief is built, updated, and used for action planning at runtime with uncertain action outcomes.

Table 5.3 reports a summary of the comparison with standard BTs and BT with dynamic expansion for *Scenarios 1* and *2*.

Table 5.3: Summary of comparison

Approach	# Hand-crafted nodes	Unforeseen contingen.	Selective adaptation	Adaptive safety guarantees
Standard BT	27	✗	✗	✓
Dynamic BT [31]	1	✓	✗	Only for the task as a whole
Ours	6	✓	✓	✓

Comparison with ROSPlan

One may argue that other solutions such as ROSPlan [26] could also be used for planning and execution in robotics in dynamic environments. ROSPlan leverages automated planning with PDDL2.1, but it is not designed for fast reaction in case of dynamic changes in the environment caused by external events, and would not work in case of a partially observable initial state. Consider *Scenario 1*. At the start of the task, the robot can sense that the gripper is empty and it has access to its current base location. However, it has no information about the state of the table (occupied or free) on which the red cube needs to be placed. At the start of the mission, the table has then 50% chance of being occupied and a 50% chance of being free, since it cannot be observed. Yet, the whole task can be planned at a high level, as in Fig. 5.9, and be executed. Once the robot reaches the placing location, observations regarding the state of the table become available and the internal beliefs can be updated until enough evidence is collected and a decision can be taken. Without knowing the full state at the start, a solution with ROSPlan

would require making assumptions on the value of the unknown states, and either planning for the worst-case scenario, which might not even be needed, or failing during execution and re-plan.

Design principles

We position our work in between two extremes, namely fully offline planning and fully online dynamic expansion of BTs. In our method, a designer can decide if to lean towards a fully offline approach or a fully online synthesis. The choice depends on the task at hand and the modeling of the actions pre- and postconditions. Even though the design of behaviors is still an art, we give some design principles that can be useful in the development of robotic applications using this hybrid BTs and Active Inference method. Take for instance Fig. 5.14 and Fig. 5.9. Prior nodes for local adaptation can be included in the behavior when there are several contingencies to consider or action preconditions to be satisfied in order to achieve a sub-goal. A designer can: 1) plan offline where the task is certain or equivalently where a small number of things can go wrong; 2) use prior nodes implemented with Active Inference to decide at runtime the actions to be executed whenever the task is uncertain. This is a compromise between a fully defined plan where the behavior of the robot is predefined in every part of the state space and a fully dynamic expansion of BTs which can result in a sub-optimal action sequence [31]. This is illustrated in Fig.5.9, where the actions for holding and placing an object are chosen online due to various possible unexpected contingencies, whereas the `moveTo` action is planned. Prior nodes should be used whenever capturing the variability of a part of a certain task would require much effort during offline planning.

5.7. Discussion

In this work, we considered a mobile manipulator in a retail store domain with a particular focus on plan execution. In this scope, offline planning is arguably better suited to offload computations at runtime for the parts of the task that do not change frequently. In our case, this was the sequence of states to stock a product that was encoded in a BT. On the other hand, at the cost of additional online computations, local online planning is better suited for execution in uncertain environments such as a busy supermarket, because one can avoid planning beforehand for every contingency. We achieved this through Active Inference. With the proposed method we can leverage the complementary advantages of both offline and online planning, and in the following, we discuss in detail the choice of using BTs and Active Inference specifically.

5.7.1. Active Inference as a planning node in a BT

We opted for the use of Active Inference with action preconditions as a planning node in the BT because this allows achieving online PDDL-style planning with noisy observations and partially observable probabilistic states. An alternative solution to our approach could be to use a simple PDDL planner in conjunction with a filtering scheme. By re-planning for a small sub-task at the same frequency used for the Active Inference node, one can achieve similar reactiveness to our approach.

However, by means of Active Inference, one can make use of the full probabilistic information on the states, and one does not require full knowledge of the symbolic initial state.

5.7.2. Why choose BTs

An experienced roboticist could also wonder why we opted for BTs in the first place, instead of other PDDL-style planning approaches to encode the solution to a task. First of all, the focus of this paper is on the runtime adaptability and reactivity in dynamic environments with partially observable initial state, and not on the generation of complex offline plans.

In this context, BTs are advantageous because they are designed to dispatch and monitor actions execution at runtime. This allows to quickly react to changes in the environment that are not necessarily a consequence of the robot's own actions. A plan that is generated through PDDL planning and executed bypassing BTs cannot provide this type of reactivity unless one defines specific re-planning strategies to mimic BTs' reactiveness.

However, we see potential extensions of this work by combining it with other PDDL planning methods. For instance, PDDL can be used to automatically generate plans for which their execution is optimized through BTs [87]. This allows for instance to take advantage of parallel action execution and would remove the need to hand-design a BT. Additionally, an action at runtime can be executed as soon as its requirements are available instead of waiting for what is established offline by the planner.

5.7.3. Why choose Active Inference

Active Inference could potentially be substituted by other valid POMDP approaches. We see two possible ways of doing so, that is either using an *offline* or an *online* POMDP solver.

First, one could solve a policy *offline* and then use it for online decision-making. This approach can be more effective than Active Inference once the transition matrices, the reward, and the task are fixed. However, the addition of new symbolic actions (so new skills as transitions), or a substantial change in the task while using the same skills, would require re-computing the policy. In addition, planning offline for all possible states and action combinations is a much larger problem than computing a plan online for the current state only. As concluded in other works [7], for the parts of a task subject to frequent unpredictable changes, computing locally an online plan as we do with Active Inference is preferable to offline policies.

Second, one could perform *online* decision making without offline computations and achieve similar performance [103, 144]. Compared to both offline and online POMDPs, however, Active Inference exposes extra model parameters to bridge abstract common sense knowledge with discrete decision-making. These extra model parameters can be updated with runtime information to adapt plans on the fly. Take for instance the prior over plans $p(\pi) = Cat(\boldsymbol{E})$. The $\boldsymbol{E}$ vector can be updated at runtime to steer the decision-making while computing the posterior distribution $\boldsymbol{\pi} = \sigma(\ln \boldsymbol{E} - \boldsymbol{G}_\pi - \boldsymbol{F}_\pi)$ [126]. This vector can be used to encode common sense and

habits [126, 62], but can also be used to adapt the plans online due for instance to runtime component failure. This opens up many possibilities for extensions, to be explored in future work. We are particularly interested in Active Inference because it is a flexible and unified framework that connects different branches of control theory at different abstraction levels. Active Inference can unify (i) abstract decision-making with guarantees, as in this paper, (ii) adaptive [109] and fault-tolerant [107, 9, 8] torque control, as well as (iii) state estimation and learning.

5.8. Conclusions

In this work, we tackled the problem of action planning and execution in real-world robotics. We addressed two open challenges, namely hierarchical deliberation, and continual online planning, by combining BTs and Active Inference. The proposed algorithm and its core idea are general and independent of the particular robot platform and task. Our solution provides local reactivity to unforeseen situations while keeping the initial plan intact. In addition, it is possible to easily add safety guarantees to override the online decision-making process thanks to the properties of BTs. We showed how robotic tasks can be described in terms of free-energy minimization, and we introduced action preconditions and conflict resolution for Active Inference by means of dynamic priors. This means that a robot can locally set its own sub-goals to resolve a local inconsistency, and then return to the initial plan specified in the BT. We performed a theoretical analysis of the convergence and robustness of the algorithm, and the effectiveness of the approach is demonstrated on two different mobile manipulators and different tasks, both in simulation and real experiments.

6

Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations

*T*HE *previous chapter tackled the problem of reactive symbolic action planning, but it still relied on hard-coded strategies for the single skills. This chapter proposes a more versatile approach for performing contact-rich skills with limited modeling effort. We achieve this by using a physics simulator as the dynamical model for Model Predictive Path Integral Control. We showcase the performance of this method in complex non-prehensile manipulation tasks and whole-body prehensile manipulation. This chapter also serves as the foundation for Chapter 7, where we combine MPPI and discrete Active Inference for task and motion planning.*

This chapter is a verbatim copy of the submitted paper [110]:

- 📄 **Corrado Pezzato***, Chadi Salmi*, Max Spahn*, Elia Trevisan*, Javier Alonso Mora, and Carlos Hernández Corbato. *"Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations"*. Submitted to IEEE Robotics and Automation Letters (2023).

* Indicates equal contribution

6.1. Abstract

We present a method for sampling-based model predictive control that makes use of a generic physics simulator as the dynamical model. In particular, we propose a Model Predictive Path Integral controller (MPPI), that uses the GPU-parallelizable IsaacGym simulator to compute the forward dynamics of a problem. By doing so, we eliminate the need for explicit encoding of robot dynamics and contacts with objects for MPPI. Since no explicit dynamic modeling is required, our method is easily extendable to different objects and robots and allows one to solve complex navigation and contact-rich tasks. We demonstrate the effectiveness of this method in several simulated and real-world settings, among which mobile navigation with collision avoidance, non-prehensile manipulation, and whole-body control for high-dimensional configuration spaces. This method is a powerful and accessible open-source tool to solve a large variety of contact-rich motion planning tasks.

Code and videos available at: https://sites.google.com/view/mppi-isaac/

6.2. Introduction

As robots become increasingly integrated into our daily lives, their ability to navigate and interact with the environment is becoming more important than ever. From collision avoidance to moving obstacles out of the way to pick up some objects, robots must be able to plan their motions while accounting for contact with their surroundings. At the same time, robotic platforms are becoming more and more complex, and include many Degrees Of Freedom (DOF) in order to achieve agile and dexterous movements. All this poses many challenging problems to motion planners, such as collision-free navigation in complex and dynamic environments, high DOF mobile manipulation, contact-rich tasks such as picking and pushing, and in-hand manipulation.

Solutions to these challenges exist but are often specialized and not easily transferable to different scenarios. Learning-based approaches, for example, can leverage physics simulators to train policies for complex tasks but require extensive training and resources. For instance, the in-hand manipulation of a cube [3] took years to develop and required 6144 CPU cores and 50 hours of training to learn a policy. On the other hand, model-based approaches like Model Predictive Control (MPC) can solve challenging tasks [121]. However, MPC often relies on constrained optimization, requiring constraint simplifications, precise modeling, and ad-hoc solutions to handle discontinuous dynamics in contact-rich tasks [64, 92]. While utilizing motion memory for warm-starting optimization can enhance performance [86, 79], the above limitations still persist.

In recent literature, Model Predictive Path Integral (MPPI) control [140] and its information-theoretic counterpart [142] addressed optimal control problems via importance sampling, mitigating challenges tied to constrained optimization algorithms dealing with non-convex constraints and discontinuous dynamics. However, substantial modeling remains necessary. In this paper, we propose a training-free model-based framework for real-time control of complex systems, where one designs
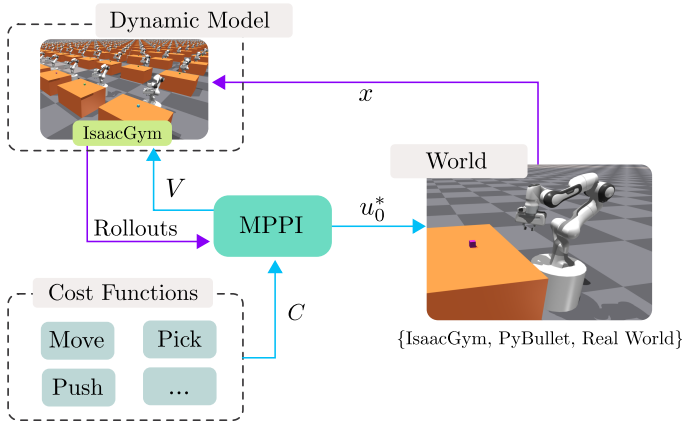
Figure 6.1: Scheme of the proposed method using IsaacGym as the dynamic model for MPPI. At each time step, IsaacGym is reset to the current world's state x, and random input sequences V are applied for the horizon T, to every environment. The resulting rolled-out trajectories are used by MPPI to approximate the optimal control u_0^* given a cost function C.

only a task cost function, not the problem's dynamics and contact models. We introduce the idea of using a general GPU-parallelizable physics simulator, Isaac-Gym [85], as the dynamic model for MPPI. By doing so, we create a robust framework that can generalize to various tasks. An overview is given in Figure 6.1.

6.2.1. Related work

This section provides an overview of selected works focusing on motion planning and contact-rich tasks in robotics. Motion planning pipelines are categorized as global and local motion planning [125]. Local motion planning encompasses approaches like operational space control, geometric methods such as Riemannian Motion Policies [81] and Optimization Fabrics [113, 130], and receding-horizon optimization formulations like Model Predictive Control (MPC) [22] that may incorporate learned components [63]. Most MPC algorithms rely on constrained optimization and assume smooth dynamics. However, contact-rich tasks pose challenges due to their non-smooth and hybrid nature, involving sticking and sliding frictions or entering contacts, requiring extensive modeling and ad-hoc solutions for pushing tasks [64, 92].

In contrast, Model Predictive Path Integral (MPPI) control [140, 142] is an approach for sampling-based MPC that approximates optimal control via parallel sampling of input sequences. MPPI is gradient-free and well-suited for systems with non-linear, non-convex, discontinuous dynamics and cost functions. It has successfully controlled high-degree-of-freedom manipulators in real-time [16], using collision-checking functions and a neural network for self-collision avoidance [36]. However, these approaches have limited interaction with the environment. Abraham and coauthors [1] proposed an ensemble MPPI, a variation that handles complex tasks and adapts to parameter uncertainty, but the task modeling remains unclear,

and no open-source implementation is available.

To alleviate the problem of explicit modeling, some studies have addressed the use of physics simulators for sampling-based MPC. For instance, Carius and coauthors [25] use the RaiSim simulator to sample waypoints for foot placement of a quadruped. Moreover, Howell et al. [65] proposed a sampling-based MPC method that employs MuJoCo [133] as a dynamic model for rolling out sampled input sequences. This offloads modeling efforts to the physics engine, simplifying controller design. However, MuJoCo's parallelization capabilities are constrained by the number of CPU threads, limiting real-time performance when many samples are required to solve a task. Moreover, results are presented only in simulation.

A high number of samples is particularly crucial in tasks such as non-prehensile manipulation with robot manipulators. Traditional approaches often involve sampling end-effector trajectories on a plane, relying on additional controllers for robot actuation and learned models for predictions [4, 33]. For instance, Arruda et al. [4] use a forward-learned model trained on 326 real robot pushes. This model is employed by an MPPI controller to plan push manipulations as end-effector trajectories. Cong et al. [33] train a Long Short-Term Memory-based model to capture push dynamics using a dataset of 300 randomized objects. End-effector trajectories are sampled within a rectangular 2D workspace. Both methods require a separate controller to convert cartesian motions into joint commands, and both perform push manipulation through a sequence of pushes, resulting in discontinuous motion. These methods are not easily transferable to other robots, particularly for non-holonomic mobile pushing.

6.2.2. Contributions

In this paper, we present a novel combination of Model Predictive Path Integral (MPPI) control with the GPU-parallelizable simulator IsaacGym to tackle a variety of contact-rich motion planning problems. While our method utilizes existing techniques, it originally integrates them. The two key contributions of this work are:

- The integration of the MPPI controller with IsaacGym, distinguishing our approach from prior works that used explicit dynamic models in MPPI. Our method facilitates collision checking and contact-rich manipulation tasks without requiring explicit modeling or learning of rigid body interactions. Our solution allows smooth real-time control of real-world high-degrees of freedom systems efficiently computing hundreds of rollouts in parallel.

- A method that can address various motion planning challenges, including collision avoidance, prehensile and non-prehensile manipulation, and whole-body control with diverse robots. We provide an open-source implementation that can be reused and extended for various robotic tasks.

We demonstrate our method in several contact-rich problems with different robotic platforms and real-world experiments, as well as comparisons against specialized baselines.

6.3. Sampling-based MPC via parallelizable physics simulations

In this section we describe the integration of MPPI with IsaacGym, which enables real-time control of complex contact-rich robotic systems with minimal modeling.

6.3.1. Background theory on MPPI

In this section, we give an overview of the background theory of MPPI. For more theoretical insights, please refer to the original publications [140, 142]. MPPI is a method to solve stochastic optimal control problems for discrete-time dynamical systems such as

$$x_{t+1} = f(x_t, v_t), \quad v_t \sim \mathcal{N}(u_t, \Sigma), \tag{6.1}$$

where the nonlinear state-transition function f describes how the state x evolves over time t with a control input v_t. MPPI samples K noisy input sequences V_k. These sequences are then applied to the system to simulate K state trajectories Q_k, $k \in [1, K]$, over a time horizon T:

$$Q_k = [x_0, f(x_0, v_0), \ldots, f(x_{T-1}, v_{T-1})]. \tag{6.2}$$

Given the state trajectories Q_k and a designed cost function C to be minimized, the total state-cost S_k of an input sequence V_k is computed by functional composition $S_k = C(Q_k)$. Then, each rollout is weighted by importance sampling weights w_k, computed via an inverse exponential of S_k with tuning parameter β, normalized by η. The minimum sampled cost $\rho = \min_k S_k$ is subtracted for numerical stability, leading to:

$$w_k = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}(S_k - \rho)\right), \quad \sum_{k=1}^{K} w_k = 1 \tag{6.3}$$

The parameter β is also known as *inverse temperature*. The weights are then used to compute the approximate optimal control input sequence U^*:

$$U^* = \sum_{k=1}^{K} w_k V_k \tag{6.4}$$

Equation (6.3) and Equation (6.4) demonstrate the approach taken to approximate the optimal control. Equation (6.4) represents a weighted average of sampled control inputs, while Equation (6.3) assigns exponentially higher weights to less costly inputs. The first input u_0^* of the sequence U^* is applied to the system. Then the process is repeated.

6.3.2. Proposed algorithm

We now describe how we use MPPI with IsaacGym, summarized in Algorithm 3. We initialize an input sequence U_{init} as a vector of zeroes with a length of T, where T is the time horizon in steps. We then sample K sequences of additive input noise $\mathcal{E}_k$ for exploring the input space around U_{init}. The key concept is that, instead of

explicitly defining a nonlinear transition function f, we use IsaacGym to compute the next state x_{t+1} given x_t and control input v_t. This is done by reading the current state of the environment, resetting the state of the simulator to the observed values, and then applying the noisy control input sequence to simulate the state trajectories in IsaacGym. Note that these K state trajectories can be computed independently of each other. We make use of this property to forward simulate all the rollouts in parallel leveraging the parallelization capabilities of IsaacGym. Instead of sampling from a Gaussian distribution, we follow the strategy of a recent paper [16] that proposes to sample *Halton Splines* instead for better exploration and smoother trajectories. Similar to the approach of Bhardwaj and coauthors [16], we calculate knot points as a Halton sequence. Given the knots, we fit a smooth spline approximation using standard Python modules and then we evaluate the spline at regular intervals to determine $\mathcal{E}_k$. Unlike Bhardwaj et al., we do not update the variance of the sampling distribution. Instead, we keep it as a tuning parameter, constant during execution. Updating the variance as in Bhardwaj et al. can lead to better convergence to a goal, but it also leads to stagnation of the control over time, which is harmful in the contact-rich tasks considered in this paper. Once the task begins, we reset our K simulation environments on IsaacGym to the current observed *world* state x. In parallel, we can now roll out the sampled input sequences V_k into state trajectories Q_k using K simulation environments on IsaacGym and compute their corresponding cost S_k using the designed *cost function* C. The cost is discounted over the planning horizon T by a factor γ [16]:

$$S_k = \sum_{t=0}^{T-1} \gamma^t C(x_{t,k}, v_{t,k}) \tag{6.5}$$

Next, we can compute the *importance sampling* weights w_k as in Equation (6.3). Note that the normalization factor η is a useful metric to monitor, as it indicates the number of samples assigned significant weights. We use this insight to automatically tune the parameter β to maintain η within an upper and lower bound:

$$\beta_{t+1} = \begin{cases} 0.9\beta & \text{if } \eta > \eta_{max} \\ 1.2\beta & \text{if } \eta < \eta_{min} \\ \beta & \text{otherwise} \end{cases} \tag{6.6}$$

Empirically, we observed in all performed tasks that setting $5 < \eta < 10$ is a good balance for smooth behavior. Finally, an approximation of the optimal control sequence U^* can now be computed via a weighted average of the sampled inputs Equation (6.4). U_{init} is now updated with U^*, time-shifted backward of one timestep so that it can be used as a warm-start for the next iteration, $U_{init} = [U_1^*, ..., U_{T-1}^*, U_{T-1}^*] \in \mathbb{R}^T$. The second last input in the shifted sequence is propagated to the last input as well. From the sequence U^*, only the first input u_0^* is applied to the system, and the next iteration starts.

Algorithm 3: Proposed Approach

1: **Initialize:**
  $U_{init} = [0, \ldots, 0] \; \{U_{init} \in \mathbb{R}^T\}$
  $\mathcal{E}_k \leftarrow sampleHaltonSplines() \; \{k = 1...K\}$
2: **while** *taskNotDone* **do**
3:   $x \leftarrow observeEnvironment()$
4:   $resetSimulations(x)$
5:   **for** $k = 1 \ldots K$ **do** {in parallel}
6:     $V_k = U_{init} + \mathcal{E}_k$
7:     $[Q_k, S_k] \leftarrow computeRolloutCost(V_k, \gamma)$
8:     $w_k \leftarrow importanceSampling$ ▷ eq. (6.3)
9:   **end for**
10:   $\beta \leftarrow updateBeta(\beta, \eta)$ ▷ eq. (6.6)
11:   $U^* = \sum_{k=1}^{K} w_k V_k$
12:   $U_{init} \leftarrow timeShift(U^*)$
13:   $applyInput(u_0^*)$
14: **end while**

6.3.3. Exploiting the physics simulator features

IsaacGym provides useful information and general models that are particularly useful for robot control in contact-rich tasks. Besides being useful to simulate the physical interaction of rigid bodies, we leverage IsaacGym for *collision checking* and *tackling model uncertainty* with domain randomization.

Collision checking

Collision checking in robotics can be challenging for a number of reasons, one of them being computational complexity. This is particularly true if the task requires continuous collision checking as the robot moves in dense environments with complex object shapes. To overcome this problem, approximations are often introduced with the convexification of the space. However, this requires several heuristics and can hinder robot motions in complex scenes.

Instead, we propose to tackle the problem of collision checking by using the already available *contact forces tensor* from IsaacGym, which is available for each simulation step. To avoid collisions, we then define a cost function proportional to the contact forces for the MPPI:

$$C_{coll} = \omega_c \sum F_{obst}, \tag{6.7}$$

where F_{obst} are the contact forces exerted on the different obstacles. This allows us to perform continuous collision checking at each time step over the horizon T, with arbitrary complex shapes. By heavily penalizing contacts with obstacles, the robot will avoid collisions. On the other hand, by relaxing the weight ω_c one can allow for certain contacts required for the task, such as rolling a ball against a wall (Section 6.4.3).

Tackling model uncertainty

IsaacGym is designed to easily support domain randomization. We use this feature to randomize the object properties in each environment in case of contact-rich tasks, such that uncertainty is incorporated in every rollout for the MPPI. Effectively, this allows to account for uncertainty in environment perception. Specifically, starting from nominal physics properties, in every rollout objects are spawned with uncertainty on mass and friction nominal values, sampled from a uniform distribution. Additionally, the object size is also randomized with additive Gaussian noise, see Section 6.4 for experiment-specific details. Therefore, every simulation is different from the others, and all simulations are different from the *world* such that we can account for model mismatch. In a sense, we perform a sort of domain randomization in real time to address the challenge of model uncertainty and imperfect perception. In the next section, we evaluate the performance of our method in different scenarios.

6.4. Experiments

We perform several experiments in three different categories: 1) *motion planning and collision avoidance*, 2) *whole-body control* of high DOF systems in contact-rich settings, and 3) *non-prehensile manipulation*. Experiments and simulations are carried out on an Alienware Laptop with Nvidia 3070 Ti graphics card. The software implementation consists of our open-source Python package that can easily be installed, tested, and extended to new robots and tasks. For the real-world tests, we used a Robot Operating System (ROS) wrapper to connect the robot to the planner, and we used a motion capture setup to determine the pose of manipulated objects.

6.4.1. Motion planning and collision avoidance

We compare the performance of the proposed method in a pure local motion planning setting, i.e. no interaction with the environment. This aims to showcase the fact that our method is comparable to state-of-the-art techniques when no contact is involved. The main focus is the quantitative analysis of the method compared to two baselines, specifically optimization fabrics [113, 130] and a simple MPC formulation solved with ForcesPro [145]. We use an already available benchmark setup, the *localPlannerBench*[129]. We present results for two cases, namely a holonomic robot, and a robotic arm (Franka Emika Panda). For all experiments, we randomize five obstacles and the goal positions in $N = 100$ runs, see Figure 6.2 for some examples. Solutions by the three methods are assessed using four metrics, e.g. time to reach the goal, path length, solver time, and minimum clearance.

The compared methods show minimal differences in path length clearance for both examples (Figure 6.3). However, our method consistently achieves faster convergence to the goal (Figures 6.3 and 6.4, and Table 6.1). This is attributed to the perfect representation of the robot's collision shapes used in our method, compared to the enclosing spheres in the ForcesPro MPC and optimization fabrics. It should be noted that our approach incurs higher computational times (Table 6.1) due to the physics simulations performed by IsaacGym.
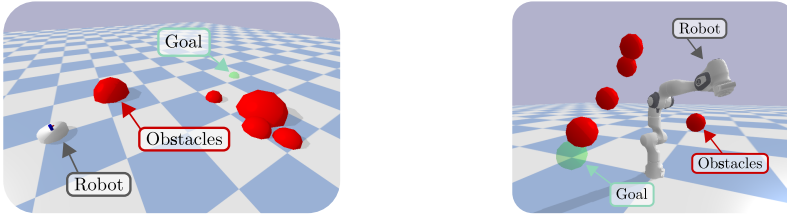
6

Figure 6.2: Examples for pure motion planning benchmark setups. Left: point robot with 3 DOF. Right: manipulator with 7 DOF.
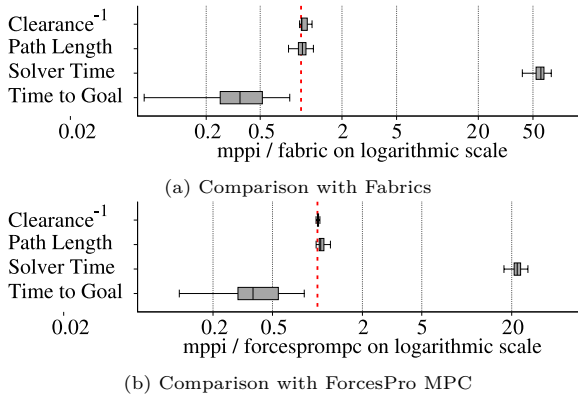

(a) Comparison with Fabrics

(b) Comparison with ForcesPro MPC

Figure 6.3: Results in pure motion planning problems for point robot with 3 DOF.

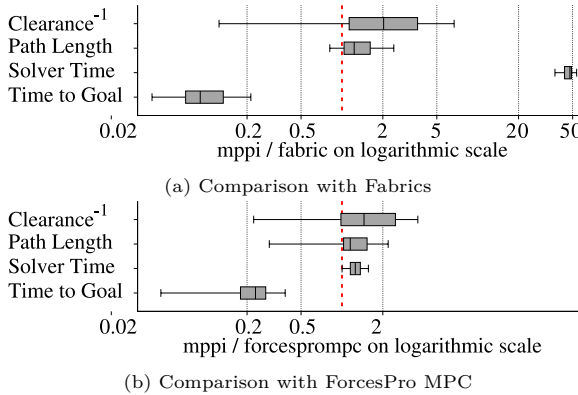(a) Comparison with Fabrics

(b) Comparison with ForcesPro MPC

Figure 6.4: Results in pure motion planning problems for a robot manipulator with 7 DOF.

Despite this, our method remains competitive in motion planning applications and offers significant advantages in contact-rich tasks, as demonstrated in the following sections.

Table 6.1: Summary of motion planning experiments

Metric	Robot	Fabric	ForcesPro MPC	MPPI
Average Solver Time	Point	**1.0**ms	2.5ms	55ms
	Panda	**1.4**ms	51ms	63ms
Average Time to Goal	Point	7.4s	6.1s	**2.7**s
	Panda	9.6s	4.2s	**0.8**s

6.4.2. Prehensile manipulation with whole-body control

Our approach scales well with the complexity of the robot. In Figure 6.5, the task is to relocate an object from a table to an $[x, y, z]$ location using a mobile manipulator with 12 DOF.
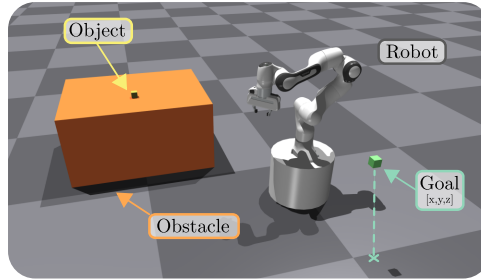
Figure 6.5: Whole-body motion of a mobile manipulator moving a cube from initial to desired location. $\omega_t = 8$, $\omega_{O_p} = 2$, $\omega_{ee} = 1$, $\omega_c = 0.1$, $T = 6$, $dt = 0.04$, $K = 500$.

Although this is arguably a complex task for a robot, which usually requires manual engineering of a sequence of movements, such as navigation to a specific base goal, and pre-post grasps, the solution is rather simple with our method. In fact, we specify the following cost function for the task:

$$C_{pick} = C_{dist} + C_{ee} + C_{coll}, \tag{6.8}$$

where we only consider the Euclidean distance of the end-effector to the object, and the object to the goal:

$$C_{dist} = \omega_t ||p_{EE} - p_O|| + \omega_{O_p} ||p_G - p_O||,$$

and we give the incentive to keep the end-effector pointing downwards, using pitch θ and roll ϕ, with $C_{ee} = \omega_{ee} ||[\phi, \theta] - [0, 0]||$. By sampling all the DOF at once, including the base and the gripper, we achieve a fluid motion from start to end with no added heuristics for pick positions. We performed 5 pick-and-deliver tasks, and the time taken was 8.19 ± 1.51s. Mass and friction are randomized with 30% uncertainty sampled uniformly. Object size is randomized with additive Gaussian noise with a standard deviation of 1mm.

For smooth whole-body motions of high DOF systems, like this one, one requires a high number of samples (i.e. a few hundred). Empirically, when the number of samples is greater than 50, a GPU pipeline is computationally cheaper than a CPU and scales better. By using IsaacGym, we can compute all the 500 samples required for mobile manipulation in parallel, computing the next control input online at $25Hz$.

6.4.3. Non-prehensile manipulation

One advantage of using a physics simulator is that one can leverage generic physics rules for contacts, thus eliminating the need for learning or engineering specialized contact models. We demonstrate this in non-prehensile manipulation tasks involving a 7-DOF arm (Figure 6.6) and two different mobile robots (Figures 6.7, 6.8 and 6.9). In Section 6.4.3, we apply our method to the two different tasks tackled in Arruda et al. [4] and Cong et al. [33] and we compare with their final results. Additionally, in Section 6.4.3, we demonstrate the ease of transferring our approach to different robots, including differential-drive.

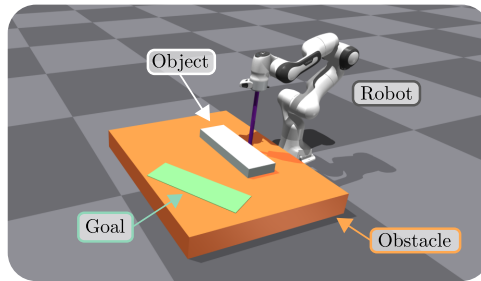**Comparison with baselines for pushing with a robot arm**

Figure 6.6: Example of non-prehensile push task with a 7-DOF robot arm using an object from Cong et al. [33].

We consider two baselines for non-prehensile pushing. In the first one, Arruda and coauthors [4] tackle the problem of pushing a relatively small object to a target pose with either 0 (Pose 1) or 90 deg (Pose 2). They also consider sequences of push actions starting far from the object. In the second baseline, Cong and coauthors [33] consider 5 relatively big objects and assume the robot's end effector is close to the object during execution. Since we do not have access to the same hardware, and the authors of the considered baselines do not provide their models and data, we only compare against their final results. Whenever their data cannot be retrieved for a certain result, we indicate this as n/a. We set up our simulation to match as close as possible the tasks in the baseline using the available information from the papers. Finally, we tune our method for the two tasks separately for a fair comparison with the individual baselines. The approach of Arruda et al. [4] utilizes an MPPI in combination with a learned model for predicting pushing effects on an object. The authors sample 2D end-effector trajectories and then rely on inverse kinematic solvers, achieving push manipulation as a sequence of disconnected

pushes. In contrast, we use MPPI to sample the control input directly as joint velocities in IsaacGym. By doing so, we achieve smooth continuous pushes where end-effector repositioning emerges naturally, and learning is not required. The cost function to be minimized for the task is:

$$C_{push} = C_{dist} + C_{push\ align} + C_{ee\ align}. \tag{6.9}$$

C_{dist} contains the weighted distance *robot-object*, and *object-goal*:

$$C_{dist} = \omega_t ||p_R - p_O|| + \omega_{O_p} ||p_G - p_O|| + \omega_{O_r} ||\psi_O - \psi_G||,$$

where p_G and ψ_G are the goal's position and orientation, while p_R and p_O denote the end-effector tip and block positions, respectively. The cost function $C_{push\ align}$ promotes keeping the object between the robot and the goal. It is computed as $cos(\alpha) + 1$, where α is the angle between the *robot-object* $(p_R - p_O)$ and *goal-object* $(p_G - p_O)$ vectors:

$$C_{push\ align} = \omega_a \Big(\underbrace{\frac{(p_R - p_O) \cdot (p_G - p_O)}{||p_R - p_O|| ||p_G - p_O||}}_{cos(\alpha)} + 1 \Big). \tag{6.10}$$

We promote the end-effector to maintain a downward orientation at height d_h using pitch θ and roll ϕ:

$$C_{ee\ align} = \omega_{ee_r} ||[\phi, \theta] - [0, 0]|| + \omega_{ee_h} ||p_{R_z} - d_h||.$$

The cost is minimized when the end-effector is close to the block at a certain height and orientation, and the block is between the end-effector and the goal at the desired goal pose[1].

We perform the same task as in Arruda et al. [4] and compare the final results of pushing a squared object on a table surface to two poses (Pose 1 and 2) with a robot arm equipped with a stick. In Table 6.2 we report our findings, showing how our method can complete the task in a fraction of the time and with double the accuracy. We used the same evaluation metric of Arruda et al. for the final cost which is a weighted average of position and orientation errors: $1.5(|p_{R_x} - p_{O_x}| + |p_{R_y} - p_{O_y}|) + 0.01|\psi_O - \psi_G|$. For every run, the manipulated object is also randomized in the same way as the rollouts. See the accompanying video to appreciate the actual behavior. We further compare our approach with Cong an coauthors [33] in terms of the success rate of non-prehensile manipulation. Particularly, we consider the same task settings, that is pushing 5 different objects to 3 different goal poses. To do so we simply change the objects in the simulation and slightly re-tune the MPPI[2].

Since the trained models from Cong et al. [33] are not provided, we only compare the final results, reported in Table 6.3. We performed 10 pushes per object, totaling 150 pushes. For the non-prehensile manipulation task with the robot arm, the mass and friction of manipulated objects have 30% uncertainty, and table friction has 90% uncertainty on the nominal value, sampled uniformly. Size is randomized with zero-mean additive Gaussian noise with a 2 mm standard deviation.

---

[1]Tuning: $\omega_t = 1$, $\omega_{O_p} = 16$, $\omega_{O_r} = 2$, $\omega_{ee_h} = 8$, $\omega_{ee_r} = 0.5$, $\omega_a = 0.8$, $dt = 0.04$, $T = 8$, $K = 500$.
[2]Tuning: $\omega_t = 5$, $\omega_{O_p} = 25$, $\omega_{O_r} = 21$, $\omega_{ee_h} = 30$, $\omega_{ee_r} = 0.3$, $\omega_a = 45$, $dt = 0.04$, $T = 8$, $K = 500$.

Table 6.2: Summary of comparison with Arruda et al. [4]

Approach	Start pose	Final cost	Time [s]
Baseline [4]	Pose 1	0.057	~ 240
	Pose 2	0.079	n/a
Ours (sim)	Pose 1	**0.029** ± 0.09	**7.65** ± 2.62
	Pose 2	**0.03** ± 0.12	**8.38** ± 2.56

Table 6.3: Summary of comparison with Cong et al. [33]

Approach	Metric	Object				
		A	B	C	D	E
Baseline [33]	Success [%]	93.5	90.9	93.9	91.6	**89.5**
	Time [s]	n/a	~ 24	~ 24	~ 24	~ 24
Ours (sim)	Success [%]	**100**	**93.3**	**96.7**	**100**	66.7
	Time [s]	**3.17** ± 1.81	**3.52** ± 1.84	**2.62** ± 1.03	**2.46** ± 0.47	**2.68** ± 1.51

By using our method, we outperformed both baselines in terms of time to completion, accuracy, and success rate, except for one manipulated object. We achieve so without limiting the sampling to 2D end-effector trajectories, without the need for learned models, and without requiring inverse kinematics solvers.

Extension to different robots
Our method is also easily extensible to different robot platforms and objects because it does not require specialized models or controllers that are robot-specific, as opposed to the baselines considered. We chose to use an omnidirectional base, and a differential drive robot, to push a box or a sphere to a goal from different initial configurations. To do so, we only need to change the environment and robot URDF in IsaacGym, and re-tune the cost function for pushing due to different hardware.

Omnidirectional push of a box The first task is the non-prehensile pushing of a box with an omnidirectional base, see Figure 6.7. Success is defined when the box is placed at the goal within 5cm in the $x - y$ direction and within 0.17 radians in rotation. The robot cannot touch obstacles. The cost function for the MPPI is the same as in Equation (6.9), re-tuned without considering end effector height and orientation since we now operate on a plane. We add an explicit term for collision avoidance $C_{coll} = \omega_c \sum F_{obst}$:

$$C_{push} = C_{dist} + C_{push\ align} + C_{coll}, \tag{6.11}$$

Omnidirectional push of a sphere One can easily extend the example above to different objects with very different dynamics. We chose a sphere instead of a box,
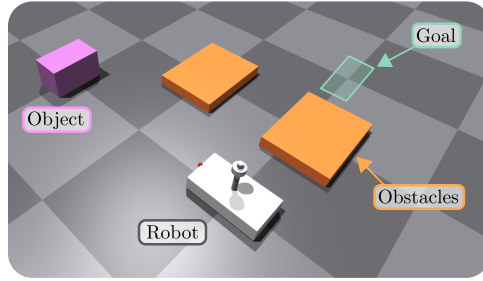
Figure 6.7: Non-prehensile task using an omnidirectional base. $\omega_t = 0.2$, $\omega_{O_p} = 2$, $\omega_{O_r} = 3$, $\omega_a = 0.6$, $\omega_c = 10$, $T = 8$, $dt = 0.04$, $K = 300$.

and we simply changed the object spawned in the simulation. For this task, we want to put the ball in between the two walls, Figure 6.8. We considered multiple runs from two different starting poses, A and B. Results are summarized in Table 6.4 and the execution can be seen in the accompanying video.
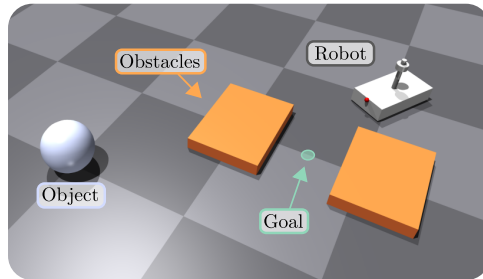

Figure 6.8: Rolling ball non-prehensile pushing. Goal: Ball placement between two obstacles. $\omega_t = 0.2$, $\omega_{O_p} = 0.1$, $\omega_{O_r} = 0$, $\omega_a = 0.1$, $\omega_c = 0.001$, $T = 8$, $dt = 0.04$, $K = 300$.

Table 6.4: Results with omnidirectional base

Object	Env.	Runs	Time [s]
Box	Pose A	5	9.66 ± 0.84
	Pose B	5	12.84 ± 0.564
Sphere	Pose A	5	8.76 ± 0.38
	Pose B	5	7.45 ± 0.59

Differential drive non-prehensile pushing We perform differential drive non-prehensile pushing, see Figure 6.9, with the same cost function as before (see Equation (6.11)) but re-tuned. One can indeed change the robot for the task by simply changing the URDF, neglecting all the additional contact modeling that would be required in a classical model-based MPC.
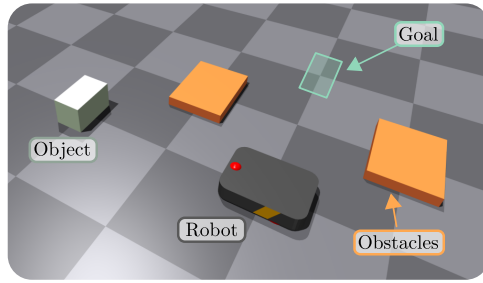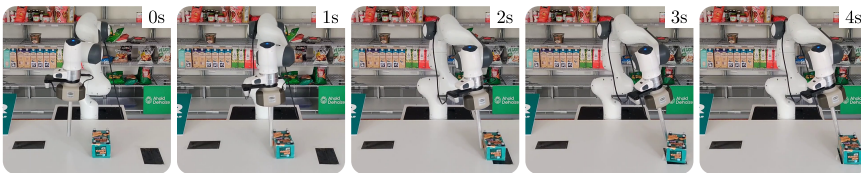
Figure 6.9: Non-prehensile differential drive pushing. Same task as the omnidirectional base. $\omega_t = 0.1$, $\omega_{O_p} = 2$, $\omega_{O_r} = 3$, $\omega_a = 0.6$, $\omega_c = 100$, $T = 12$, $dt = 0.04$, $K = 400$.

(a) Pushing straight to the goal on the right.

(b) Pushing to the goal on the left with 90° rotation.

Figure 6.10: Pushing to two different goals with a 7 DOF manipulator directly controlling all joint velocities. Our method allows the re-positioning of the end-effector around the object without specifying any particular desired contact point.

The time taken to push the box to the goal was 18.31s. In the mobile non-prehensile pushing experiments, objects to manipulate are spawned with 30% uncertainty on mass and friction sampled uniformly, while object size is randomized with Gaussian noise with a standard deviation of 5mm.

6.4.4. Real-world experiments

To demonstrate the applicability of our approach, we transfer to the real world a subset of the non-prehensile manipulation tasks previously presented in Section 6.4.3 with both the robot manipulator and the omnidirectional base. In particular, in Figure 6.10, we show the results of the 7 DOF manipulator pushing a product to two different goals, similar to the simulations corresponding to Table 6.2. As presented in Figure 6.1, the samples are rolled out in $K = 500$ simulated environments in IsaacGym which, at each timestep, are initialized to the state of the real world. Based on this, the optimal control is estimated and applied to the real system.

When transferring to the real world, compared to the experiments in Section 6.4.3, only the weights of the cost function were re-tuned. The horizon, control frequency, number of samples, structure of the cost function, and randomization of the sampled environments remained unchanged. From the experimental evaluation on the real robot, we observe that the time to complete the different pushing tasks and the final position error are comparable to the results in the simulation from Table 6.2. Importantly, these results are achieved without taking assumptions about specific contact points. Thus, the robot can naturally re-position itself and change contact location autonomously. Additionally, our method allows sampling joint velocities directly thus we do not restrict the sampling to 2D end-effector trajectories to be tracked by another controller as often seen in other approaches. Lastly, to demonstrate robustness, we disturb the execution of pushing tasks with the manipulator and the omnidirectional base by hand. Since we do not assume the robot to be behind the object to be pushed for successful execution, and since the planning and execution happen in real-time at $25Hz$, we can largely perturb the task and let the robot compensate.

Figure 6.11: Qualitative real-world experiments with disturbances. The behavior of the controlled system can be best appreciated in the accompanying video.

6.5. Discussion

In this section, we discuss key aspects and potential future work related to our solution. First, the computational demands of planning and control with our method can be high when extending the time horizon to several seconds. To keep the time horizon limited for real-time control while preventing being trapped in local minima, future work should incorporate global planning techniques such as A*, RRT, and Probabilistic Roadmaps (PRM) [74] to guide the local planner. Similarly to warm starting predictive controllers [86, 79], one could make use of motion libraries of previous executions or learned policies along with random rollouts, to improve the sampling efficiency and exploration. Second, in real-world scenarios, uncertainties and discrepancies between simulated and actual environments could present challenges for achieving precise movements and manipulation. We utilized randomization of object properties in the rollouts to address some uncertainties. However, online system identification to converge to the true model parameters is not performed. Enhancing the robustness of the MPPI algorithm itself by reducing

model uncertainty, as demonstrated by Abraham and coauthors [1], could further improve performance. Third, tuning control algorithms for optimal performance is time-consuming. However, implementing autotuning techniques can automate the process and reduce manual effort. Finally, incorporating additional sensor support such as lidars and signed distance fields could be highly beneficial. This would enable, for instance, collision avoidance without the need for complex simulated worlds.

6.6. Conclusions

We presented a way to perform Model Predictive Path Integral controller (MPPI) that uses a physics simulator as the dynamic model. By leveraging the GPU-parallelizable IsaacGym simulator for parallel sampling of forward trajectories, we have eliminated the need to explicitly encode robot dynamics, contacts, and rigid-body interactions for MPPI. This makes our method easily adaptable to different objects and robots for various contact-rich motion-planning tasks. Through a series of simulations and real-world experiments, we have demonstrated the effectiveness of this approach in various scenarios, including motion planning with collision avoidance, non-prehensile manipulation, and whole-body control. We showed how our method can compete with state-of-the-art motion planners in case of no interactions, and how it outperforms by a margin other approaches for contact-rich tasks. In addition, we provided an open-source implementation that can be used to reproduce the presented results, and that can be adapted to new tasks and robots.

6

7

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

*T*HIS *chapter builds upon Chapter 5 and Chapter 6, blending the high-level reactive behavior obtained through Active Inference, with the powerful motion planning capabilities of MPPI for contact-rich tasks. The key idea is that Active Inference is now scheduling costs to be minimized instead of symbolic actions, allowing the use of MPPI for long-term tasks. This leads to a scheme capable of performing long-term missions in a reactive manner. Thanks to the use of parallel sampling with MPPI, we can evaluate and blend different strategies planned by Active Inference into one coherent and smooth behavior. We demonstrate the potential of this idea both in simulation and in the real world.*

7.1. Abstract

Task and Motion Planning (TAMP) has made strides in complex manipulation tasks, yet the execution robustness of the planned solutions remains overlooked. In this work, we propose a method for reactive TAMP to cope with runtime uncertainties and disturbances. We combine an Active Inference planner (AIP) for adaptive high-level action selection and a novel Multi-Modal Model Predictive Path Integral controller (M3P2I) for low-level control. This results in a scheme that simultaneously adapts both high-level actions and low-level motions. The AIP generates alternative symbolic plans, each linked to a cost function for M3P2I. The latter employs a physics simulator for diverse trajectory rollouts, deriving optimal control by weighing the different samples according to their cost. This idea enables blending different robot skills for fluid and reactive plan execution, accommodating plan adjustments at both the high and low levels to cope, for instance, with dynamic obstacles or disturbances that invalidate the current plan. We have tested our approach in simulations and real-world scenarios. Experiments' videos are available here.

7.2. Introduction

Task and motion planning is a powerful class of methods for solving complex long-term manipulation problems where logic and geometric variables influence each other. TAMP [55] has been successfully applied to domains such as table rearrangement, stacking blocks, or solving the Hanoi tower. Despite impressive recent results [101], the environments in which TAMP is executed are usually not dynamic, and the plan is executed in open-loop [56]. Recent works [134, 90, 82] recognized the importance of robustifying the execution of TAMP plans to be able to carry them out in the real world reliably. However, they either rely only on the adaptation of the action sequence in a plan [108, 27, 31, 82] or only on the motion planning problem in a dynamic environment given a fixed plan [134, 90]. This paper aims to achieve reactive execution by simultaneously adapting high-level actions and low-level motions.

A key challenge in reactive TAMP is handling geometric constraints that might not be known at planning time. For instance, moving a block to a desired location may necessitate pulling rather than pushing if the block is situated in a corner. Planning at the high level for these contingencies is hard especially when not assuming full knowledge of the scene at planning time. Another challenging scenario could be a pick-and-place task with dynamic obstacles and human disturbances. Different grasping poses are necessary for diverse locations of the objects and obstacles. Thus, a TAMP algorithm should be able to adapt to these configurations on the fly.

We address these challenges by proposing a control scheme that jointly achieves reactive action selection and robust low-level motion planning during execution. We propose a high-level planner capable of providing alternative actions to achieve a goal. These actions are translated to different cost functions for our new Multi-Modal Model Predictive Path Integral controller for motion planning. This motion planner leverages a physics simulator to sample parallel motion plans that minimize

the given costs and computes one coherent control input that effectively blends different strategies. To achieve this, we build upon two of our recent works: 1) an Active Inference planner (AIP) [108] for symbolic action selection, and 2) a Model Predictive Path Integral (MPPI) controller [110] for motion planning. We extend the previous AIP to plan possible alternative plans, and we propose a new Multi-Modal Model Predictive Path Integral controller (M3P2I) that can sample in parallel these alternatives and smoothly blend them.

7.2.1. Related work

To robustly operate in dynamic environments, reactive motion planners are necessary. Toussaint and coauthors [134] provide a reactive MPC strategy to execute a TAMP plan as a given linear sequence of constraints. Instead of composing primitive skills, they derive the control law from a composition of constraints for MPC. The reactive nature of the approach allows coping with disturbances and dynamic collision avoidance during the execution of a TAMP plan. The work of Migimatsu and Bohg [90] formulates a TAMP plan in object-centric Cartesian coordinates, showing how this allows coping with perturbations such as moving a target location. However, the two works above [134, 90] do not consider adaptation at the symbolic action level if a perturbation invalidates the current plan.

Several papers focused on adapting and repairing high-level action sequences during execution. Paxton et al. [106] propose representing robot task plans as robust logical-dynamical systems. The method can effectively adapt the logic plan to deal with external human disturbances. Similarly, Harris et al. [60] presented a method to coordinate control chains and achieve robust plan execution through plan switching and controller selection. A recent paper [66] proposes to use Monte Carlo Tree Search in combination with Isaac Gym to speed up task planning for multi-step object retrieval from clutter, where complex physical interaction is required. This is a promising direction, but the method only allows for high-level reasoning while executing pre-defined motions in an open loop. Works from Zhou et al. [148] and Li et al. [82] combined behavior trees and linear temporal logic to adapt the high-level plan to cope with cooperative or adversarial human operators, environment changes, or failures. In our previous work [108], Active Inference and behavior trees were combined to provide reactive action selection in long-term tasks in partially observable and dynamic environments. This method achieved hierarchical deliberation and continual online planning, making it particularly appealing for the problem of reactive TAMP at hand. In this paper, we extend our previous method [108] by bridging the gap to low-level reactive control by planning cost functions instead of symbolic actions.

At the lower level, Model Predictive Control (MPC) is a widely used approach [15, 123, 128]. However, most MPC methods rely on convexifying constraints and cost functions. Notably, manipulation tasks often involve discontinuous contacts that are hard to differentiate.

Sampling-based MPCs offer a promising alternative, such as MPPI [143, 16]. These algorithms can handle non-linearities, non-convexities, or discontinuities of the dynamics and costs. MPPI relies on sampling control input sequences and for-

ward system dynamics simulation. The resulting trajectories are weighted according to their cost to approximate an optimal control input. Abraham and coauthors [1] propose an ensemble MPPI to cope with model parameters uncertainty. Sampling-based MPCs are generally applied for single-skill execution, such as pushing or reaching a target point. Howell and coauthors [65] pointed out for future work that one could use a high-level agent to set the cost functions for the sampling-based MPC for long-horizon cognitive tasks. We follow this line of thought and propose a method to compose cost functions for long-horizon tasks reactively. Moreover, classical MPPI approaches can only keep track of one cost function at a time. This means the task planner should propose a single plan to solve the task. However, some tasks might present geometric ambiguities for which multiple plans could be effective, and selecting what strategy to pursue can only be determined by the motion planner based on the geometry of the problem.

7.2.2. Contributions
The contribution is a reactive task and motion planning algorithm based on:

- A new Multi-Modal MPPI (M3P2I) capable of sampling in parallel plan alternatives to achieve a goal, evaluating them against different costs. This enables the smooth blending of alternative solutions into a coherent behavior instead of switching based on heuristics.

- An enhanced Active Inference planner capable of generating alternative cost functions for M3P2I.

7.3. Background
In this section, we present the background knowledge about the Active Inference planner and Model Predictive Path Integral Control to understand the contributions of this paper. We refer the interested reader to the original articles [108, 140, 142] for a more in-depth understanding of the single techniques.

7.3.1. Active Inference Planner
The AIP is a high-level decision-making algorithm that relies on symbolic states, observations, and actions [108]. Each independent set of states in AIP is a factor, and the planner contains a total of n_f factors. For a generic factor f_j where $j \in \mathcal{J} = \{1, ..., n_f\}$, it holds:

$$s^{(f_j)} = \left[s^{(f_j,1)}, s^{(f_j,2)}, ..., s^{(f_j,m^{(f_j)})} \right]^\top,$$
$$\mathcal{S} = \left\{ s^{(f_j)} | j \in \mathcal{J} \right\} \tag{7.1}$$

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

The continuous state of the world $x \in \mathcal{X}$ is discretized through a symbolic observer such that the AIP can use it. Discretized observations o are used to build a

probabilistic belief about the symbolic current state. Assuming one set of observations per state factor with $r^{(f_j)}$ possible values:

$$o^{(f_j)} = \left[o^{(f_j,1)}, o^{(f_j,2)}, ..., o^{(f_j,r^{(f_j)})}\right]^{\top},$$
$$\mathcal{O} = \left\{o^{(f_j)} | j \in \mathcal{J}\right\} \tag{7.2}$$

The robot has a set of symbolic actions that can act then their corresponding state factor:

$$a_\tau \in \alpha^{(f_j)} = \left\{a^{(f_j,1)}, a^{(f_j,2)}, ..., a^{(f_j,k^{(f_j)})}\right\},$$
$$\mathcal{A} = \left\{\alpha^{(f_j)} | j \in \mathcal{J}\right\} \tag{7.3}$$

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

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

where $\texttt{prec}_{a^{(f_j,\cdot)}}$ and $\texttt{post}_{a^{(f_j,\cdot)}}$ are *first-order logic predicates* that can be evaluated at run-time. A logical predicate is a boolean-valued function $\mathcal{B} : \mathcal{X} \to \{\texttt{true}, \texttt{false}\}$. Finally, we define the logical state $l^{(f_j)}$ as a one-hot encoding of $s^{(f_j)}$. The AIP computes the posterior distribution over p plans $\boldsymbol{\pi}$ through free-energy minimization [108]. The symbolic action to be executed by a robot in the next time step is the first action of the most likely plan, denoted with $\pi_{\zeta,0}$:

$$\zeta = \max(\underbrace{[\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_p]}_{\boldsymbol{\pi}^\top}), \ a_{\tau=0} = \pi_{\zeta,0}. \tag{7.4}$$

7.3.2. Model Predictive Path Integral Control

MPPI is a method for solving optimal stochastic problems in a sampling-based fashion [140, 142]. Let us consider the following discrete-time systems:

$$x_{t+1} = f(x_t, v_t), \quad v_t \sim \mathcal{N}(u_t, \Sigma), \tag{7.5}$$

where f, a nonlinear state-transition function, describes how the state x evolves over time t with a control input v_t. u_t and Σ are the commanded input and the variance, respectively. K noisy input sequences V_k are sampled and then applied to the system to forward simulate K state trajectories Q_k, $k \in [0, K-1]$, over a time horizon T. Given the state trajectories Q_k and a designed cost function C to be minimized, the total state-cost S_k of an input sequence V_k is computed by evaluating $S_k = C(Q_k)$. Finally, each rollout is weighted by the importance sampling weights w_k. These are computed through an inverse exponential of the cost S_k with tuning parameter β and normalized by η. For numerical stability, the minimum sampled

cost $\rho = \min_k S_k$ is subtracted, leading to:

$$w_k = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}(S_k - \rho)\right), \quad \sum_{k=1}^{K} w_k = 1 \tag{7.6}$$

The parameter β is called *inverse temperature*. The importance sampling weights are finally used to approximate the optimal control input sequence U^*:

$$U^* = \sum_{k=1}^{K} w_k V_k \tag{7.7}$$

The first input u_0^* of the sequence U^* is applied to the system, and the process is repeated. At the next iteration, U^* is used as a warm-start, time-shifted backward of one timestep. Specifically, the second last input in the shifted sequence is also propagated to the last input. In this work, we build on top of our previous MPPI approach [110], where we employed IsaacGym as a dynamic model of the system to forward simulate trajectory rollouts.

7.4. Methodology

The proposed method is depicted in Figure 7.1. After a general overview, we discuss the three main parts of the scheme: *action planner*, *motion planner*, and *plan interface*.
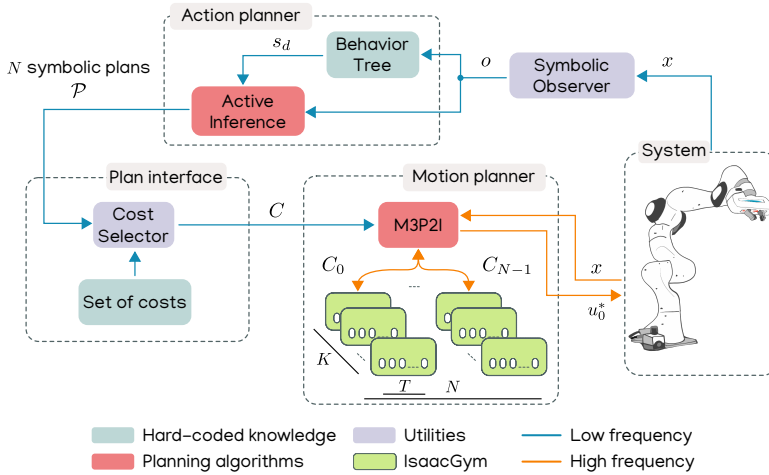

Figure 7.1: Proposed control scheme. Given symbolic observations o of the environment, the action planner computes N different plan alternatives that are linked to individual cost functions C_i for M3P2I. The latter samples control input sequences and uses an importance sampling scheme to approximate the optimal control u_0^*.

7.4.1. Overview

The proposed scheme works as follows. First, the *symbolic observers* translates continuous states x into discretized symbolic observations o. These are passed to a

BT and Active Inference. The BT encodes the skeleton solution of a task in terms of desired states instead of actions as previous work [108]. The running node in a BT sets the current desired state s_d for Active Inference. The latter computes N alternative symbolic plans based on the current symbolic state and the available symbolic actions. The symbolic actions are encoded as action templates with pre-post conditions that Active Inference uses to construct action sequences to achieve the desired state. After the plans are generated, the plan interface links the first action $a_{0,i}$, $i = 0...N - 1$ of each plan to a cost function C_i. The cost functions are sent to M3P2I, which samples $N \cdot K$ different control input sequences. The input sequences are forward simulated using IsaacGym, which encodes the dynamics of the problem [110]. The resulting trajectories are evaluated against their respective costs. Finally, an importance sampling scheme calculates the approximate optimal control u_0^*. All processes are running continuously during execution at different frequencies. The action planner runs, for instance, at $1Hz$ while the motion planner runs at $25Hz$. An overview can be found in Algorithm 4.

Algorithm 4: Overview of the method

1: **while** Task not completed **do**
2:   $o \leftarrow GetSymbolicObservation$
3:   /* Get current desired state from BT */
4:   $AIP.s_d \leftarrow BT$
5:   /* Get current action plans from Active Inference */
6:   $\mathcal{P} \leftarrow AIP.parell\_act\_sel(o)$ ▷ Algorithm 5
7:   /* Translate action plan to cost function */
8:   $C \leftarrow Interface(\mathcal{P})$
9:   /* Compute motion commands */
10:   $M3P2I.command(C)$ ▷ Algorithm 7
11: **end while**

7.4.2. Action planner - Active Inference

The action planner we propose is a modified version of our previous method [108] to allow Active Inference to plan alternative action sequences to achieve a desired state. In contrast to our previous work [108] where only one action a_τ for the next time step is computed, we modify the AIP to generate action alternatives. In particular, instead of stopping the search for a plan when a valid executable action a_τ is found, we repeat the search while removing that same a_τ from the available action set $\mathcal{A}$. This simple change is effective because we are looking for alternative actions to be applied at the next step, and the AIP builds plans backward from the desired state [108]. The pseudocode is reported in Algorithm 5. The algorithm will cease when no new actions are found, returning a list of possible plans $\mathcal{P}$. This planner is later integrated with the parallel motion planner M3P2I to evaluate different alternatives in real time. This increases the robot's autonomy and performance, at the same time reducing the number of heuristics to be encoded in the action planner.

Algorithm 5: Generate alternative plans using Active Inference

---

1: $a_\tau \leftarrow AIP.act\_sel(o)$ ▷ from our past work [108]
2: Set $\mathcal{P} \leftarrow \emptyset$
3: /* *Look for alternatives* */
4: **while** $a_\tau ! = \text{none}$ **do**
5:     $\mathcal{P}.append(a_\tau)$
6:     $\mathcal{A} = \mathcal{A}\backslash\{a_\tau\}$
7:     $a_\tau \leftarrow AIP.act\_sel(o)$ ▷ from our past work [108]
8: **end while**
9: **Return** $\mathcal{P}$

7.4.3. Motion planner - Multi-Modal MPPI (M3P2I)

We propose a Multi-Modal MPPI capable of sampling different plan alternatives from the AIP. Traditional MPPI approaches consider *one* cost function and *one* sampling distribution. In this work, we propose keeping track of N separate control input sequences corresponding to N different plan alternatives/costs. This is advantageous because it offers a general approach to exploring different strategies in parallel. We perform N separate sets of importance weights, one for each alternative, and only ultimately, we combine the weighted control inputs in one coherent control. This allows the smooth blending of different strategies. Assume we consider N alternative plans, a total of $N \cdot K$ samples. Assume the cost of plan $i, i \in [0, N)$ to be formulated as:

$$S_i(V_k) = \sum_{t=0}^{T-1} \gamma^t C_i(x_{t,k}, v_{t,k}) \tag{7.8}$$

$\forall k \in \kappa(i)$ where $\kappa(i)$ is the integer set of indexes ranging from $i \cdot K$ to $(i+1) \cdot K - 1$. State $x_{t,k}$ and control input $v_{t,k}$ are indexed based on the time t and trajectory k. The random control sequence $V_k = [v_{0,k}, v_{1,k}, \ldots, v_{T-1,k}]$ defines the control inputs for trajectory k over a time horizon T. The trajectory $Q_i(V_k) = [x_{0,k}, x_{1,k}, \ldots, x_{T-1,k}]$ is determined by the control sequence V_k and the initial state $x_{0,k}$. C_i is the cost function for plan i. Finally, $\gamma \in [0, 1]$ is a discount factor that evaluates the importance of accumulated future costs. As in classical MPPI approaches, given the costs $S_i(V_k)$, we can compute the importance sampling weights associated with each alternative as:

$$\omega_i(V_k) = \frac{1}{\eta_i} \exp\left(-\frac{1}{\beta_i}\left(S_i(V_k) - \rho_i\right)\right), \forall k \in \kappa(i) \tag{7.9}$$

$$\eta_i = \sum_{k \in \kappa(i)} \exp\left(-\frac{1}{\beta_i}\left(S_i(V_k) - \rho_i\right)\right) \tag{7.10}$$

$$\rho_i = \min_{k \in \kappa(i)} S_i(V_k) \tag{7.11}$$

We use the insight from our past work [110] to 1) sample Halton splines instead of Gaussian noise for smoother behavior, 2) automatically tune the inverse temperature

β_i to maintain the normalization factor η_i within certain bounds. The latter is helpful since η_i indicates the number of samples to which significant weights are assigned. If η_i is close to the number of samples K, an unweighted average of sampled trajectories will be taken. If η_i is close to 1, then the best trajectory sample will be taken. We observed that keeping η_i between 5% and 10% of K generates smooth trajectories. As opposed to our previous method [110], we update η *within a rollout* to stay within bounds instead of updating it once per iteration, see Algorithm 6. We use μ_i to denote the action sequence of plan i over a time horizon

---

Algorithm 6: Update inverse temperature β_i

1: **while** $\eta_i \notin [\eta_l, \eta_u]$ **do**
2:     $\rho_i \leftarrow \min_{k \in \kappa(i)} S_i(V_k)$;   ▷ Equation (7.11)
3:     $\eta_i \leftarrow \sum_{k \in \kappa(i)} \exp(-\frac{S_i(V_k) - \rho_i}{\beta_i})$;   ▷ Equation (7.10)
4:     **if** $\eta_i > \eta_u$ **then**   ▷ bigger than upper bound
5:         $\beta_i = 0.9 * \beta_i$
6:     **else if** $\eta_i < \eta_l$ **then**   ▷ smaller than lower bound
7:         $\beta_i = 1.2 * \beta_i$
8:     **end if**
9: **end while**
10: **Return** ρ_i, η_i, β_i

$\mu_i = [\mu_{i,0}, \mu_{i,1}, \ldots, \mu_{i,T-1}]$. Each sequence is weighted by the corresponding weights leading to:

$$\mu_i = \sum_{k \in \kappa(i)} \omega_i(V_k) V_k \tag{7.12}$$

At every iteration, we add to μ_i the sampled noise from *Halton splines* [16]. Then, we forward simulate the state trajectories $Q_i(V_k)$ using IsaacGym as we previously did [110]. Finally, given the state trajectories corresponding to the plan alternatives, we need to compute the weights and mean for the overall control sequence. To do so, we concatenate the N state-costs $S_i(V_k), i \in [0, N)$ and represent it as $\tilde{S}(V)$. Therefore, we calculate the weights for the whole control sequence as [16]:

$$\tilde{\omega}(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}\left(\tilde{S}(V) - \rho\right)\right) \tag{7.13}$$

Similarly, η, ρ are computed as in Equations (7.10) and (7.11) but considering $\tilde{S}(V)$ instead. The overall mean action over time horizon T is denoted as $u = [u_0, u_1, \ldots, u_{T-1}]$. For each timestep t:

$$u_t = (1 - \alpha_u)u_{t-1} + \alpha_u \sum_{k=0}^{N \cdot K - 1} \tilde{\omega}_k(V) v_{t,k} \tag{7.14}$$

where α_u is the step size that regularizes the current solution to be close to the previous u_{t-1}. The optimal control is set to $u_0^* = u_0$. Note that through Equation (7.13), we can smoothly fuse different strategies to achieve a goal in a general way.

Algorithm 7: Multi-Modal Model Predictive Path Integral Control (M3P2I)

1: **Parameters:** N, K, T;
2: **Initialize:** $\mu_i = \mathbf{0}, u = \mathbf{0}, \in \mathbb{R}^T \ \forall i \in [0, N)$
3: **while** task not completed **do**
4:    $x \leftarrow GetStateEstimate()$;
5:    $InitIsaacGym(x)$
6:    /* Begin parallel sampling of alternatives */
7:    **for** $i = 0$ to $N - 1$ **do**
8:      **for** $k \in \kappa(i)$ **do**
9:        $S_i(V_k) \leftarrow 0$;
10:       Sample noise $\mathcal{E}_k \leftarrow SampleHaltonSplines()$;
11:       $\mu_i \leftarrow BackShift(\mu_i)$;
12:       **for** $t = 0$ to $T - 1$ **do**
13:         $Q_i(V_k) \leftarrow ComputeTrajIsaacGym(\mu_i + \mathcal{E}_k)$
14:         $S_i(V_k) \leftarrow UpdateCost(Q_i(V_k))$;   ▷ Equation (7.8)
15:       **end for**
16:      **end for**
17:    **end for**
18:    /* Begin computing trajectory weights */
19:    **for** $i = 0$ to $N - 1$ **do**
20:      $\rho_i, \eta_i, \beta_i \leftarrow UpdateInvTemp(i)$;   ▷ Algorithm 6
21:      $\omega_i(k) \leftarrow \frac{1}{\eta_i} \exp(-\frac{1}{\beta_i}(S_i(V_k) - \rho_i)), \forall k$;   ▷ Equation (7.9)
22:      $\mu_i = \sum_{k \in \kappa(i)} \omega_i(V_k) V_k$ ▷ Equation (7.12)
23:    **end for**
24:    /* Begin control update */
25:    $\tilde{\omega}(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}\left(\tilde{S}(V) - \rho\right)\right)$ ▷ Equation (7.13)
26:    **for** $t = 0$ to $T - 1$ **do**
27:      $u_t = (1 - \alpha_u)u_{t-1} + \alpha_u \sum_{j=0}^{N \cdot K - 1} \tilde{\omega}_k v_{t,k}$ ▷ Equation (7.14)
28:    **end for**
29:    $ExecuteCommand(u_0^* = u_0)$
30:    $u = BackShift(u)$
31: **end while**

The pseudocode is summarized in Algorithm 7. After the initialization, we sample Halton splines and forward simulate the plan alternatives using IsaacGym to compute the costs (Lines 7-17). The costs are then used to update the weights for each plan and update their means (Lines 19-23). Finally, the mean of the overall action sequence is updated (Line 27), and the first action from the mean is executed.

7.4.4. Plan interface

The plan interface is a component that takes the possible alternative symbolic actions in $\mathcal{P}$ and links them to their corresponding cost functions, forwarding the latter to M3P2I. For every symbolic action a robot can perform, we store a cost function in a database that we can query at runtime, bridging the output of the action planner to the motion planner.

7.5. Experiments

We evaluate the performance of our method in two different scenarios. The first is a *pull-push scenario* for non-prehensile manipulation of an object with an omnidirectional robot capable of both pulling and pushing. The second is the *object stacking scenario* with a 7-DOF manipulator with dynamic obstacles and external disturbances at runtime.

7.5.1. Push-pull scenario

This scenario is depicted in Figure 7.2. One object has to be placed to a goal, situated in one of the corners of an arena. The object to be pushed can have different initial locations, for instance, in the middle of the arena or on one of the corners. There are also static and dynamic obstacles, and the robot can push or pull the object. We define the following action templates for the AIP and the cost functions for M3P2I.
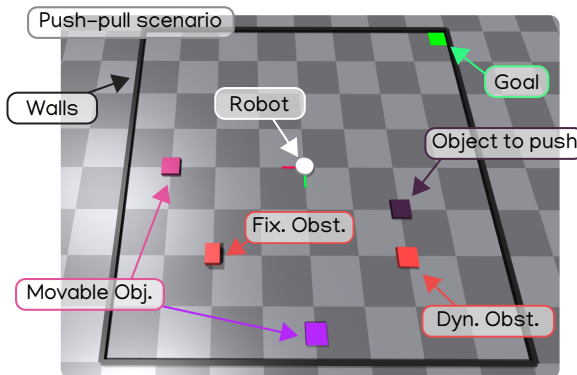
Figure 7.2: Push-pull scenarios. The dark purple object has to be placed on the green area. The robot can pull or push the object while avoiding dynamic and fixed obstacles. The objects and goals can have different initial positions.

Action templates for AIP

The AIP for this task requires one state $s^{(goal)}$ and a relative symbolic observation $o^{(goal)}$ that indicates when an object is at the goal. This is defined as:

$$s^{(goal)} = \begin{bmatrix} \texttt{isAt(goal)} \\ \texttt{!isAt(goal)} \end{bmatrix}, o^{(goal)}) = \begin{cases} 0, ||p_G - p_O|| \leq \delta \\ 1, ||p_G - p_O|| > \delta \end{cases} \tag{7.15}$$

Actions	Preconditions	Postconditions
push(obj,goal)	-	$l^{(goal)} = [1\ 0]^\top$
pull(obj,goal)	-	$l^{(goal)} = [1\ 0]^\top$

where p_G, p_O represent the positions of the goal and the object in a 3D coordinate system. δ is a constant threshold determined by the user. The mobile robot can either push, pull, or move. These skills are encoded in the action planner as follows:

The post-condition of the action push(obj,goal) is that the object is at the goal, similarly for the pull action. Note that we do not add complex heuristics to encode the geometric relations in the task planner to determine when to push or pull; instead, we will exploit parallel sampling in the motion planner later. The BT for this task sets s_d as a preference for $l^{(goal)} = [1\ 0]^\top$. The BT would contain more desired states for pushing or pulling several blocks. Our approach can be extended to multiple objects in different locations, for instance, and accommodate more involved pre-post conditions and fallbacks since it has the same properties as our past method [108].

Cost functions for MPPI

We need to specify a cost for each symbolic action. The cost function for pushing object O to the goal G is defined as:

$$C_{push}(R,O,G) = C_{dist}(R,O) + C_{dist}(O,G) + C_{ori}(O,G) \\ + C_{align\_push}(R,O,G) \tag{7.16}$$

where minimizing $C_{dist}(O,G) = \omega_{dist} \cdot ||p_G - p_O||$ makes the object O close to the goal G. $C_{ori}(O,G) = \omega_{ori} \cdot \phi(\Sigma_O, \Sigma_G)$ defines the orientation cost between the object O and goal G. We define ϕ for symmetric objects as:

$$\phi(\Sigma_u, \Sigma_v) = \min_{i,j \in \{1,2,3\}} (2 - ||\vec{u}_1 \cdot \vec{v}_i|| - ||\vec{u}_2 \cdot \vec{v}_j||) \tag{7.17}$$

where $\Sigma_u = \{\vec{u}_1, \vec{u}_2, \vec{u}_3\}, \Sigma_v = \{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ form the orthogonal bases of two coordinates systems. Minimizing this cost makes two axes in the coordinate systems of the object and goal coincide.

The align cost $C_{align\_push}(R,O,G)$ is defined as:

$$C_{align\_push}(R,O,G) = \omega_{align\_push} \cdot h(cos(\theta)), \tag{7.18}$$

$$cos(\theta) = \frac{(p_R - p_O) \cdot (p_G - p_O)}{||p_R - p_O|| \cdot ||p_G - p_O||}, \tag{7.19}$$

$$h(cos(\theta)) = \begin{cases} 0, & cos(\theta) \leq 0 \\ cos(\theta), & cos(\theta) > 0 \end{cases} \tag{7.20}$$

This makes the object O lie at the center of robot R and goal G so that the robot can push it, as illustrated in Figure 7.3.

Similarly, the cost function of making the robot R pull object O to the goal G can be formulated as:
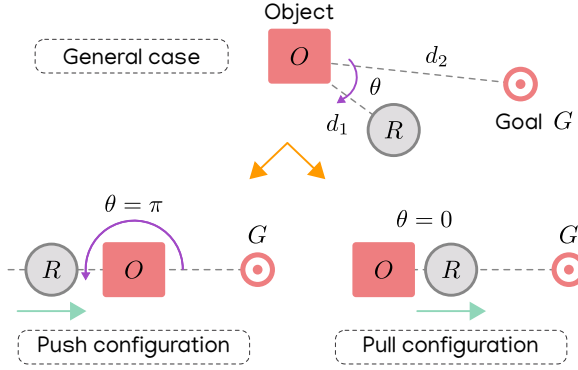
Figure 7.3: Push and pull ideal configurations. The robot R has to push or pull the object O to the goal G.

$$C_{pull}(R,O,G) = C_{dist}(R,O) + C_{dist}(O,G) + C_{ori}(O,G)$$
$$+ C_{align\_pull}(R,O,G) + C_{act\_pull}(R,O,G) \tag{7.21}$$

where the align cost $C_{align\_pull}(R,O,G)$ makes the robot R lie between the object O and goal G, see Figure 7.3. While pulling, we simulate a suction force in IsaacGym, and we are only allowed to sample control inputs that move away from the object. We enforce this through $C_{act\_pull}(R,O,G)$. Mathematically:

$$C_{align\_pull}(R,O,G) = \omega_{align\_pull} \cdot h(-cos(\theta)) \tag{7.22}$$

$$C_{act\_pull}(R,O,G) = \omega_{act\_pull} \cdot h(\frac{(p_O - p_R) \cdot \vec{u}}{||p_O - p_R|| \cdot ||\vec{u}||}) \tag{7.23}$$

An example execution of push and pull can be seen in Figure 7.4. Finally, we consider an additional cost $C_{dyn\_obs}(R,D)$ to avoid collisions with (dynamic) obstacles while operating. We use a constant velocity model to predict the position of the dynamic obstacle D in the coming horizon and try to maximize the distance between the latter and the robot:

$$C_{dyn\_obs}(R,D) = \omega_{dyn\_obs} \cdot e^{-||p_R - p_{D_{pred}}||} \tag{7.24}$$

where $p_{D_{pred}}$ is the predicted position of dynamic obstacle.

Results

We test the performance of our multi-modal MPPI approach in two configurations of the arena: a) The object is in the middle of the arena, and the goal is to one corner, and b) both the object and the goals are in different corners. For each arena configuration, we test three cases: the robot can either only push, only pull, or combine the two through our M3P2I. At the beginning of the task, the AIP plans for the two alternatives, pushing and pulling, and forwards the solution to the plan
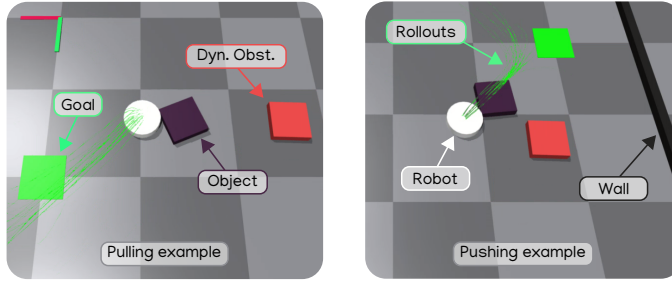
Figure 7.4: Illustrative example of pulling and pushing a block to a goal. The strategy differs according to the object, goal location, and dynamic obstacle position. What action to perform is decided at runtime through multi-modal sampling.

interface. Then, the M3P2I starts minimizing the costs until the AIP observes the completion of the task. We performed 20 trials per case, per arena configuration, for a total of 120 simulations. By only pulling an object, the robot cannot tightly place it on top of the goal in the corner; on the other hand, by only pushing, the robot cannot retrieve the object from the corner. By blending the push and pull actions, we can complete the task in every tested configuration. Table 7.1 shows that the multi-modal case outperforms push and pull in both arena configurations. It presents lower position and orientation errors and a shorter completion time. The behavior is best appreciated in the accompanying video.

Table 7.1: Simulation Results for Push and Pull scenario

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

7.5.2. Object stacking scenario

We now illustrate how we integrated the features of M3P2I with AIP's reactivity, enabling adjustments in both low-level motions and high-level actions. We address the challenge of stacking objects (Figure 7.5) with external task disruptions, necessitating adaptive actions like re-grasping with different pick configurations (e.g., top or side picking). We showcase the robot's ability to rectify plans by repeating actions or compensating for unplanned occurrences, such as unexpected obstacles obstructing the path. We benchmark our approach's performance against the cube-stacking task outlined by Makoviychuk and coauthors [85]. Our experiments employ a 7-DOF manipulator and include both simulation and real-world settings.
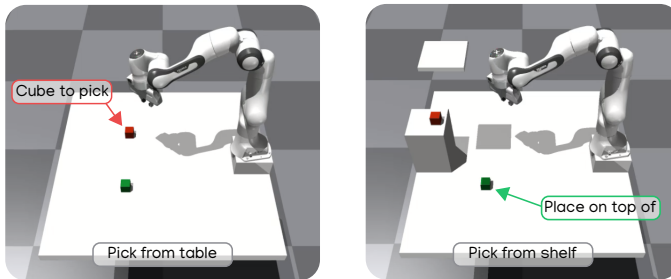
Figure 7.5: Pick-place scenarios. The red cube has to be placed on top of the green cube. The red cube can be either on the table or a constrained shelf, requiring different pick strategies from the top or the side, respectively.

Action templates for AIP

For this task, we define the following states $s^{(reach)}$, $s^{(hold)}$, $s^{(preplace)}$, $s^{(placed)}$, and their corresponding symbolic observations. The robot has four symbolic actions, summarized in Table 7.2. The symbolic observers to estimate the states are defined

Table 7.2: Actions with their Pre and Post conditions

Actions	Preconditions	Postconditions
reach(obj)	–	$l^{(reach)} = [1\ 0]^\top$
pick(obj)	reachable(obj)	$l^{(hold)} = [1\ 0]^\top$
prePlace(obj)	holding(obj)	$l^{(preplace)} = [1\ 0]^\top$
place(obj)	atPreplace(obj)	$l^{(placed)} = [1\ 0]^\top$

as follows. To estimate whether the gripper is close enough to the cube, we define the relative observation $o^{(reach)}$. We set $o^{(reach)} = 0$ if $\delta_r \le 0.012$, where $\delta_r = ||p_{ee} - p_O||$ measures the distance between the end effector ee and the object O. $o^{(reach)} = 1$ otherwise. To estimate whether the robot is holding the cube, we define:

$$o^{(hold)} = \begin{cases} 0, \delta_f < 0.065 \text{ and } \delta_f > 0.058 \\ 1, \delta_f \ge 0.065 \text{ or } \delta_f \le 0.058 \end{cases} \tag{7.25}$$

where $\delta_f = ||p_{ee\_l} - p_{ee\_r}||$ measures the distance between the two gripper's fingers. To estimate whether the cube reaches the pre-place location, we define:

$$o^{(preplace)} = \begin{cases} 0, C_{dist}(O, P) < 0.01 \text{ and } C_{ori}(O, P) < 0.01 \\ 1, C_{dist}(O, P) \geq 0.01 \text{ or } C_{ori}(O, P) \geq 0.01 \end{cases} \tag{7.26}$$

where $C_{dist}(O, P)$ and $C_{ori}(O, P)$ measure the distance and the orientation between the object O and the pre-place location P as in Equation (7.16). The pre-place location is a few centimeters higher than the target cube location, directly on top of the green cube. We use the same logic as Equation (7.26) for $o^{(placed)}$ where the place location is directly on top of the cube location. The BT for this task sets the desired state to be $s_d \rightarrow s^{(placed)} = [1\ 0]^T$, meaning the cube is correctly placed on top of the other. Note that in more complex scenarios, such as rearranging many cubes, the BT can guide the AIP as we demonstrated in past work [108].

Cost functions for M3P2I
At the motion planning level, the cost functions for the four actions are:

$$C_{reach}(ee, O, \psi) = \omega_{reach} \cdot ||p_{ee} - p_O|| \\ + \omega_{tilt} \cdot \left(\frac{||\vec{z}_{ee} \cdot \vec{z}_O||}{||\vec{z}_{ee}|| \cdot ||\vec{z}_O||} - \psi) \right) \tag{7.27}$$

$$C_{pick}(ee) = \omega_{gripper} \cdot l_{gripper} \tag{7.28}$$

$$C_{preplace}(O, P) = C_{dist}(O, P) + C_{ori}(O, P) \tag{7.29}$$

$$C_{place}(O, P) = \omega_{gripper} \cdot (1 - l_{gripper}) \tag{7.30}$$

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

Results - reactive pick and place
We first consider the pick-and-place under disturbances. We model disturbances by changing the position of the cubes at any time. We compare the performance of our method with the off-the-shelf RL method [85]. This is a readily available Actor-Critic RL example from IsaacGym, which considers the same tabletop configuration and robot arm. We compare the methods in a *vanilla* task without disturbances and a *reactive* task with disturbances. It should be noticed that the cube-stacking task from Makoviychuk et al. [85] only considers moving the cube on top of the other cube while neglecting the action of opening the gripper and releasing the cube. In contrast, our method exhibits fluent transitions between pick and place and shows robustness to interferences during the long-horizon task execution. The results can be found in Table 7.3.

We performed 50 trials per case. Even though the RL agent presents a slightly lower position error in the vanilla case, our method outperforms it in the reactive task.

Table 7.3: Simulation Results of Reactive Pick and Place

Task	Method	Training Epochs	Mean(std) pos error
Vanilla	Ours	0	0.0075 (0.0036)
	RL	1500	**0.0042** (0.0019)
Reactive	Ours	0	**0.0117** (0.0166)
	RL	1500	0.0246 (0.0960)

Results: multi-modal grasping

In this case, we consider grasping the object with different grasping poses by sampling two alternatives in parallel. That is, pick from the top or the side to cover the cases when the object is on the table or on the constrained shelf with an obstacle above. To do so, we use the proposed M3P2I and incorporate the cost functions of $C_{reach}(ee, O, \psi = 0)$ and $C_{reach}(ee, O, \psi = 1)$ as shown in Equation (7.27). This allows for a smooth transition between op and side grasp according to the geometry of the problem, see Figure 7.6.
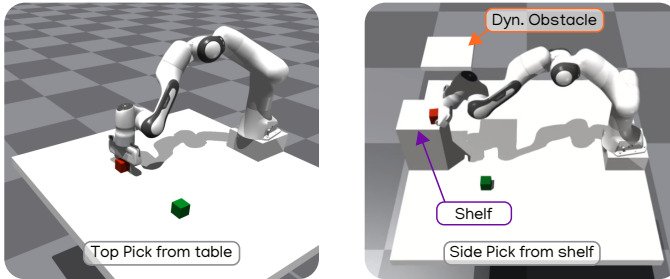
Figure 7.6: Example of different picking strategies computed by our multi-modal MPPI. The obstacle on top of the shelf can be moved, simulating a dynamic obstacle. See the accompanying video for the actual execution.

Results - Real-world experiments

We validate our method in the real world for reactive pick-and-place as shown in Figure 7.7. We consider dynamic obstacles with a stick to be avoided and other human disturbances such as moving, rotating, and stealing the cube from the gripper. M3P2I enables smooth execution and recovery while being able to grasp the cube in different configurations.

7.6. Conclusions

In this paper, to address the runtime geometric uncertainties and disturbances, we proposed a method to combine the adaptability of an Active Inference planner for high-level action selection with a novel Multi-Modal Model Predictive Path Integral Controller (M3P2I) for low-level control. We modified Active Inference to generate
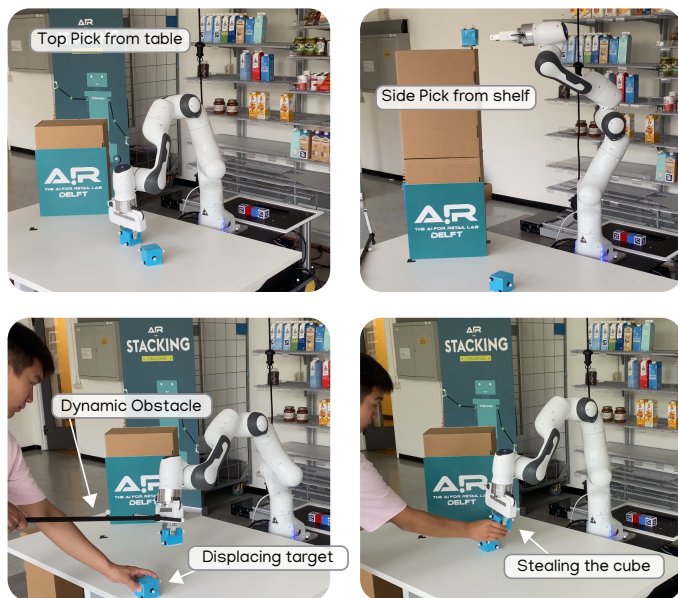
Figure 7.7: Real-world experiments of picking a cube from the table or the shelf while avoiding dynamic obstacles and recovering from task disturbances. See the accompanying video for the actual execution.

plan alternatives that are linked to costs for M3P2I. The motion planner can sample the plan alternatives in parallel, and it computes the control input for the robot by smoothly blending different strategies. In a push-pull task, we demonstrated how our proposed framework can blend both push and pull actions, allowing it to deal with corner cases where approaches only using a single plan fail. With a simulated manipulator, we showed our method outperforming a reinforcement learning baseline when the environment is disturbed while requiring no training. Simulated and real-world experiments demonstrated how our approach solves reactive object stacking tasks with a manipulator subject to severe disturbances and various scene configurations that require different grasp strategies.

8

Conclusion and discussion

*T*HIS *concluding chapter summarises the contributions of this thesis and provides an extensive discussion of the findings, pointing out fundamental questions and future directions to be explored.*

> *There is no real ending.*
> *It's just the place where you stop the story.*

Frank Herbert

8.1. Conclusions

In this work, we have contributed new control algorithms and ideas towards adaptive and reactive robots at various levels of abstraction. Specifically, we explored the application of Active Inference for low-level robot control and task planning and the utilization of MPPI for contact-rich motion planning. Through the analysis, extension, modification, and combination of Active Inference and MPPI, this thesis has advanced the state-of-the-art in terms of *low-level adaptive* and *fault-tolerant control*, *reactive high-level action planning*, and *contact-rich task and motion planning*. The proposed theories have been mathematically formalized and verified through extensive simulations and experiments on real robots. In the following, we provide a more detailed summary of the content of each contributing chapter, highlighting the specific contributions and advancements made in each area.

1. ***Low-level adaptive and fault-tolerant control with unbiased Active Inference***: Chapters 3 and 4 presented a modified version of Active Inference that explicitly incorporates control actions into the free-energy computation. This formulation offered two key advantages. Firstly, it eliminated the bias from state estimation that was present in the standard Active Inference controller. Secondly, it simplified the definition of control actions by removing the need for computing partial derivatives of free-energy. The advantages of the u-AIC have been combined with established fault detection techniques with probabilistic robustness in Chapter 3. This effectively reduced false positives compared to standard Active Inference in a simulated 2-DOF arm subject to sensory faults. This work served as the basis for Chapter 4, where the concept of unbiased Active Inference has been mathematically formalized and tested in real-world settings. This chapter established a comprehensive framework that extended beyond its application in fault-tolerant control and touched upon reference tracking and collision behaviors. The theoretical findings have been validated through experiments on a real 7-DOF robot arm, marking the first implementation of the u-AIC on an actual robot. Chapter 4 also marked the end of the exploration of low-level control during this thesis.

2. ***Reactive high-level decision making***: Chapter 5 was centered around reactive high-level action planning and task execution, shifting the focus to discrete decision-making problems. The chapter presented a novel combination of the discrete formulation of Active Inference with behavior trees. This integration has enabled the merging of offline planning with online decision-making, resulting in a fast and reactive scheme suitable for long-term tasks. In contrast to traditional approaches that plan actions to be executed, this approach utilized behavior trees to encode desired states to be achieved. These desired states serve as a guide for the online search of an action sequence using Active Inference. Notably, this work represented a pioneering application of discrete Active Inference on a real robot. Particularly, it has been applied for retail store tasks such as stocking shelves or collecting orders. This chapter provided high-level adaptive behavior but assumed a set of atomic skills to be given. These skills were pre-programmed routines that made limited use of

geometric and physical information about the environment. These turned out to be crucial for enhanced robot autonomy with less hard-coded engineering effort, as explained in the subsequent Chapter 6.

3. ***Contact-rich motion planning***: Chapter 6 leveraged object geometries and generic physics to achieve smoother execution of atomic skills without relying on extensive heuristics. This chapter explored the use of Model Predictive Path Integral Control to leverage general physics for motion planning in contact-rich scenarios. In particular, it presented the combination of MPPI with a physics simulator, IsaacGym, to be used as the dynamic model for planning. This allowed parallel sampling of forward trajectories using the simulator, eliminating the need for explicit modeling of complex robot dynamics, object interactions, and collision checking. The proposed method demonstrated applicability to various tasks, including motion planning with collision avoidance, non-prehensile manipulation, and prehensile whole-body control. Experimental results showcased competitive performance with state-of-the-art motion planners in non-interacting scenarios while outperforming other approaches in contact-rich tasks. Of particular interest is the emergence of smooth pick behaviors with mobile manipulators without manually encoding any pre or post-grasp constraints. This chapter laid the foundations for the work presented in the subsequent Chapter 7.

4. ***Reactive task and motion planning***: Chapter 7 combined the advancements of Chapter 5 and Chapter 6 into a control architecture for reactive task and motion planning. This architecture combined the Active Inference planner with model predictive path integral control. The main idea was to use Active Inference to plan cost functions instead of symbolic actions. Additionally, Active Inference has been modified to plan different (parallel) alternatives to achieve a goal. Then, the chapter proposed a Multi-Modal MPPI, the M3P2I, capable of sampling in parallel these alternatives, weighing them against their relative cost to compute one coherent optimal control input. By doing so, a robot can blend different skills without needing articulated heuristics at the task planning level. This method allowed us to achieve reactive high-level and low-level behavior, resulting in a reactive TAMP framework. The simulation results and the real-world experiments demonstrated the benefit of this method, especially in tasks that can only be solved if different skills are sampled and blended together. This chapter is the last contribution of this thesis but opens up many possibilities for the future development of autonomous robots capable of intelligently interacting with their surroundings.

8.2. Discussion

This dissertation embarked on a journey to explore robotic algorithms aiming at achieving human-level capabilities for long-term tasks in dynamic environments. While we have introduced numerous advancements in robot autonomy across various abstraction levels, we acknowledge that we have not yet unlocked human-level intelligence or dexterity. This section provides a thorough exploration, examining

both the successes and challenges faced. It offers insights into the techniques utilized and explains the reasoning behind the choices made in the contributing chapters. Finally, we outline possible directions for future work that could be addressed with the theories presented in this thesis.

8.2.1. Active Inference or MPPI? Choices and insights

At its core, this thesis aims to investigate the theories of Active Inference and MPPI for robot control. In Chapter 2, we provided the theoretical background for these two theories from a mathematical perspective. In this sub-section, instead, we will critically assess the strengths and limitations of these two approaches in relation to each other, explaining why the selection of either one or the other was appropriate for different abstraction levels within a robot control architecture. It is crucial to recognize that while Active Inference and MPPI are both connected to the idea of free-energy, the definition of the free-energy itself differs in these two theories, and is based on substantially different assumptions. Active Inference is rooted in biological assumptions, asserting that organisms seek to minimize the surprise of sensory information to ensure survival. In contrast, MPPI adopts a different form of free-energy, which refers to a mathematical quantity of a controlled system that takes the same form of Helmholtz free-energy. This disregards biological plausibility and focuses solely on deriving an optimal control strategy. These distinct assumptions have notable implications at each level of abstraction in the robot control architecture.

Low-level control and motion planning

In the context of low-level control, Active Inference demonstrated its efficacy in Chapters 3 and 4, offering an overall control law that does not necessitate a detailed model of the controlled system. This follows from the biological assumptions behind Active Inference, which forces the evolution of a dynamic system to follow a desired behavior, for instance, the one of a first-order linear system [107, 109]. This is advantageous when dealing with uncertain robot dynamics, enabling compliant control and accurate point-to-point motions in generic manipulators. In contexts where a model is not readily available, MPPI's requirement for a well-defined system description could be a limitation. Conversely, incorporating collision avoidance or discontinuous contacts within the Active Inference formulation proved challenging. This is because defining a general differentiable generative model for these cases becomes very complex rather quickly. These complexities motivated the adoption of MPPI for motion planning in contact-rich scenarios, which excels in dealing with discontinuous dynamics and arbitrary cost functions. For the sort of tasks considered in Chapter 6, such as non-prehensile manipulation, accurate models of object interactions are required. Instead of analytically defining such models or learning specialized networks to approximate contact behaviors, we proposed using a physics simulator for planning. The simple idea of leveraging the simulator itself for modeling contact dynamics opened doors to solving various contact-rich tasks that would have been challenging under the biological assumptions of Active Inference.

Importantly, it should be emphasized that Active Inference and MPPI can be

combined for low-level control as well. In some of our practical real-world experiments for contact-rich motion planning with collision avoidance, we used MPPI to provide joint velocity references to the unbiased Active Inference to achieve compliant low-level motions. Employing a fast low-level controller that abstracted the system's dynamics like u-AIC, along with a more advanced motion planner for reference generation like MPPI, led to effective and compliant robot motions.

High-level decision making

Moving on to the higher levels of the control hierarchy, the discrete formulation of Active Inference provided a nice framework for adaptive decision-making. This is because of the intuitive idea behind it, that is, a system seeks to observe the prescribed desired states. This led to our intuition of encoding desired states in a BT to guide the high-level action selection process with Active Inference, which would have been more complex with MPPI. The nature of abstract reasoning required for task planning is captured well with categorical distributions, directly aligning with Active Inference's capabilities. Classical MPPI theory, instead, only considers Gaussian distributions. Ultimately, we opted to connect Active Inference with behavior trees and MPPI through cost function scheduling, combining their complementary strengths. By doing so, we achieved a high degree of autonomy in mobile manipulators, enabling adaptive task planning, and geometric and physical reasoning for contact-rich tasks. However, our exploration is not exhaustive, and robotics still presents numerous challenges. While we addressed several hurdles in this work, areas remain open for future exploration and advancements. The field of robotics, in fact, continues to pose exciting challenges, encouraging us to continue our pursuit of creating truly human-level capable machines.

8.2.2. Challenges and future work

Why does my robot fail in such a straightforward task? This resonating question has often crossed the mind of the author during the course of this thesis, and arguably the mind of every roboticist at least once. The simple answer could be that we often demand too much from our robots while providing them with inadequate resources. In robotics, we aspire to achieve human-level autonomy in complex and unstructured environments, yet we typically equip our robots with limited sensing, reasoning, and motor skills. Below, we report what we consider important challenges and directions for future work in different areas.

Perception

Throughout this work, we focused primarily on aspects other than perception, considering it as a given. However, especially in the conducted real-world experiments, the significance of **fast and accurate perception** systems cannot be overlooked. This is particularly crucial in the case of contact-rich tasks, where future work could focus on integrating multimodal perception pipelines that incorporate visual and tactile feedback for better state estimation. To obviate the need to have accurate 3D representation in a simulator for collision avoidance with MPPI, future work could integrate methods such as signed distance fields directly in the cost

function. Additionally, online system identification could be included to enhance model accuracy. By integrating these advancements into the methodologies presented in Chapters 6 and 7, one could better bridge the reality gap and facilitate smoother transferability from simulation to real-world applications. Additionally, recent trends in robotics consider the problem of active perception. We believe this is an exciting direction for future exploration that can enhance the autonomy of robots in environments that are hard to perceive accurately.

Controllers fusion
During the development of the work in Chapter 6 [110], we briefly experimented with the addition of prior controllers, such as PID or dynamic fabrics [130], along with random samples. Although we did not include this in the thesis, this idea can potentially improve sample efficiency by providing informative samples based on designed controllers. Thus, one can extend our MPPI approach [110] to roll out in parallel classical controllers or learned policies, together with random samples to improve the overall performance of the MPPI motion planner. This would de-facto enable **controller fusion**, blending different strategies into one coherent optimal control input. MPPI could take suggestions from such controllers, steering the commanded input towards the one that works best in the current scenario while avoiding high-cost trajectories. By doing so one could again smoothly transition between different policies without the need for specific heuristics. The task and motion planning scheme in Chapter 7 could be bootstrapped to plan both costs and a set of possible prior controllers that can minimize that cost. Thus, one could have an ever-growing and more capable system where the controller sets can be augmented over time. Ultimately, as the system remains controlled solely by the output of MPPI, this setup appears scalable and readily capable of continuously incorporating advancements in learning-based skills. By combining task-specific costs with "barrier costs", one could safely deploy learned policies while preserving certain safety measures such as collision or joint limit avoidance.

Goals and costs definition
The approach of planning cost functions to be minimized, as demonstrated in Chapter 7, showcased its effectiveness for tasks where a distance measure can represent the objective. However, challenges arise when defining a cost function for more complex tasks, such as tying someone's shoe. In the realm of future research, **exploring alternative methods of specifying cost functions and goals** holds a vast potential. One avenue of exploration could involve representing goals as images [20], where planning sub-images becomes a crucial process. This approach may be particularly relevant for tasks that lack a straightforward analytical description. Moreover, taking a step further, leveraging neural networks as cost functions for MPPI can open up new possibilities to tackle tasks that are hard to describe analytically. Finally, there are also great opportunities for conditioning the cost function based on formal methods and reasoning. One could integrate common sense knowledge to smoothly steer the behavior of a robot to guarantee certain quality and safety attributes. Thus, one could change not only what a robot has to perform, but also how.

Long-term vision: towards machines with feelings
To enhance robustness in decision-making at a high level, future research could explore the concept of robots experiencing frustration or annoyance when repeatedly failing, similar to humans. Currently, robots lack the capacity for such emotional responses, which can lead to getting stuck in repetitive, ineffective behaviors. To address this, the work in Chapter 5 with behavior trees and Active Inference could **incorporate principled methods to adjust model parameters dynamically**, allowing the robot to experiment with different strategies when faced with failures. Introducing a sense of frustration could prompt the robot to explore alternative approaches, and ideally learn from its mistakes. In the same way, robots' physical constraints such as finite battery capacity, could influence their behavior. By regulating their actions to move slower when "feeling" tired due to low battery, robots could demonstrate a level of self-awareness and adaptive behavior.

The notion of feelings may be intimately connected to **the core business of consciousness**, as explored in the book "*The Hidden Spring*" by Mark Solms [127]. This raises intriguing questions about whether machines with feelings could be the ultimate path toward achieving general artificial intelligence. However, *should we even embark on the journey to develop machines with feelings*? This contemplation places both a weighty responsibility and a rewarding opportunity on researchers as it could open up new frontiers in the realms of artificial intelligence, paving the road to an exciting future of autonomous machines.

8

Bibliography

[1] Ian Abraham, Ankur Handa, Nathan Ratliff, Kendall Lowrey, Todd D. Murphey, and Dieter Fox. "Model-Based Generalization Under Parameter Uncertainty Using Path Integral Control". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 2864–2871. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2972836. (Visited on 06/19/2023).

[2] Arslan Ahmed Amin and Khalid Mahmood Hasan. "A review of Fault Tolerant Control Systems: Advancements and applications". en. In: *Measurement* 143 (Sept. 2019), pp. 58–68. ISSN: 02632241. DOI: 10.1016/j.measurement.2019.04.083. URL: https://linkinghub.elsevier.com/retrieve/pii/S0263224119304099 (visited on 08/09/2023).

[3] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. "Learning dexterous in-hand manipulation". en. In: *The International Journal of Robotics Research* 39.1 (Jan. 2020), pp. 3–20. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364919887447. URL: http://journals.sagepub.com/doi/10.1177/0278364919887447 (visited on 06/19/2023).

[4] Ermano Arruda, Michael J. Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L. Wyatt. "Uncertainty averse pushing with model predictive path integral control". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)* (2017). URL: https://www.academia.edu/35477194/Uncertainty_Averse_Pushing_with_Model_Predictive_Path_Integral_Control (visited on 06/19/2023).

[5] M. Baioumy, M. Mattamala, and N. Hawes. "Variational inference for predictive and reactive controllers". English. In: (2020). URL: https://ora.ox.ac.uk/objects/uuid:b869c8e6-ee88-494c-b4d4-df5f3c5c2c2e (visited on 06/16/2023).

[6] Mohamed Baioumy, Paul Duckworth, Bruno Lacerda, and Nick Hawes. "Active Inference for Integrated State-Estimation, Control, and Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 4665–4671. DOI: 10.1109/ICRA48506.2021.9562009.

[7]   Mohamed Baioumy, Bruno Lacerda, Paul Duckworth, and Nick Hawes. "On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference". en. In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Communications in Computer and Information Science. Cham: Springer International Publishing, 2021, pp. 819–829. ISBN: 978-3-030-93736-2. DOI: 10.1007/978-3-030-93736-2_58.

[8]   Mohamed Baioumy, Corrado Pezzato, Carlos Hernández Corbato, Nick Hawes, and Riccardo Ferrari. "Towards Stochastic Fault-Tolerant Control Using Precision Learning and Active Inference". en. In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Communications in Computer and Information Science. Cham: Springer International Publishing, 2021, pp. 681–691. ISBN: 978-3-030-93736-2. DOI: 10.1007/978-3-030-93736-2_48.

[9]   Mohamed Baioumy, Corrado Pezzato, Riccardo Ferrari, Carlos Hernández Corbato, and Nick Hawes. "Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference". In: *2021 European Control Conference (ECC)*. June 2021, pp. 1119–1125. DOI: 10.23919/ECC54610.2021.9654913.

[10]   Mohamed Baioumy, Corrado Pezzato, Riccardo Ferrari, and Nick Hawes. "Unbiased Active Inference for Classical Control". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 12787–12794. DOI: 10.1109/IROS47612.2022.9981095.

[11]   Manuel Baltieri and Christopher Buckley. "PID Control as a Process of Active Inference with Linear Generative Models". en. In: *Entropy* 21.3 (Mar. 2019), p. 257. ISSN: 1099-4300. DOI: 10.3390/e21030257. URL: https://www.mdpi.com/1099-4300/21/3/257 (visited on 06/16/2023).

[12]   Manuel Baltieri and Christopher L. Buckley. "A Probabilistic Interpretation of PID Controllers Using Active Inference". en. In: *From Animals to Animats 15*. Ed. by Poramate Manoonpong, Jørgen Christian Larsen, Xiaofeng Xiong, John Hallam, and Jochen Triesch. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 15–26. ISBN: 978-3-319-97628-0. DOI: 10.1007/978-3-319-97628-0_2.

[13]   Manuel Baltieri and Christopher L. Buckley. "On Kalman-Bucy filters, linear quadratic control and active inference". In: (2020). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2005.06269. URL: https://arxiv.org/abs/2005.06269 (visited on 06/16/2023).

[14]   Manuel Baltieri and Christopher L. Buckley. "The modularity of action and perception revisited using control theory and active inference". In: *The 2018 Conference on Artificial Life*. arXiv:1806.02649 [q-bio]. 2018, pp. 121–128. DOI: 10.1162/isal_a_00031. URL: http://arxiv.org/abs/1806.02649 (visited on 06/16/2023).

8

[15] Moses Bangura and Robert Mahony. "Real-time Model Predictive Control for Quadrotors". en. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 11773–11780. ISSN: 14746670. DOI: 10.3182/20140824-6-ZA-1003.00203. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667016434890 (visited on 08/25/2023).

[16] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. "STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation". en. In: *Proceedings of the 5th Conference on Robot Learning.* ISSN: 2640-3498. PMLR, Jan. 2022, pp. 750–759. URL: https://proceedings.mlr.press/v164/bhardwaj22a.html (visited on 06/19/2023).

[17] Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and J. Schröder. *Diagnosis and Fault-Tolerant Control.* eng. 2., ed. Berlin: Springer Berlin, 2006. ISBN: 978-3-540-35653-0.

[18] Morten Rufus Blas and Mogens Blanke. "Stereo vision with texture learning for fault-tolerant automatic baling". en. In: *Computers and Electronics in Agriculture* 75.1 (Jan. 2011), pp. 159–168. ISSN: 01681699. DOI: 10.1016/j.compag.2010.10.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S016816991000222X (visited on 06/19/2023).

[19] Rafal Bogacz. "A tutorial on the free-energy framework for modelling perception and learning". en. In: *Journal of Mathematical Psychology.* Model-based Cognitive Neuroscience 76 (Feb. 2017), pp. 198–211. ISSN: 0022-2496. DOI: 10.1016/j.jmp.2015.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0022249615000759 (visited on 06/15/2023).

[20] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. *RoboCat: A Self-Improving Foundation Agent for Robotic Manipulation.* arXiv:2306.11706 [cs]. June 2023. DOI: 10.48550/arXiv.2306.11706. URL: http://arxiv.org/abs/2306.11706 (visited on 09/12/2023).

[21] Christopher L. Buckley, Chang Sub Kim, Simon McGregor, and Anil K. Seth. "The free energy principle for action and perception: A mathematical review". en. In: *Journal of Mathematical Psychology* 81 (Dec. 2017), pp. 55–79. ISSN: 0022-2496. DOI: 10.1016/j.jmp.2017.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0022249617300962 (visited on 06/15/2023).

8

[22] Giovanni Buizza Avanzini, Andrea Maria Zanchettin, and Paolo Rocco. "Constrained model predictive control for mobile robotic manipulators". en. In: *Robotica* 36.1 (Jan. 2018), pp. 19–38. ISSN: 0263-5747, 1469-8668. DOI: 10. 1017/S0263574717000133. URL: https://www.cambridge.org/core/product/identifier/S0263574717000133/type/journal_article (visited on 06/19/2023).

[23] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. "Sequential Composition of Dynamically Dexterous Robot Behaviors". en. In: *The International Journal of Robotics Research* 18.6 (June 1999). Publisher: SAGE Publications Ltd STM, pp. 534–555. ISSN: 0278-3649. DOI: 10.1177/02783649922066385. URL: https://doi.org/10.1177/02783649922066385 (visited on 06/15/2023).

[24] Luca Massimiliano Capisani, Antonella Ferrara, and Pierluigi Pisu. "Sliding mode observers for vision-based fault detection, isolation and identification in robot manipulators". In: *Proceedings of the 2010 American Control Conference*. Baltimore, MD: IEEE, June 2010, pp. 4540–4545. DOI: 10.1109/ACC.2010.5530865. URL: http://ieeexplore.ieee.org/document/5530865/ (visited on 06/19/2023).

[25] Jan Carius, René Ranftl, Farbod Farshidian, and Marco Hutter. "Constrained stochastic optimal control with learned importance sampling: A path integral approach". In: *The International Journal of Robotics Research* 41.2 (Feb. 2022), pp. 189–209. ISSN: 0278-3649, 1741-3176. URL: http://journals.sagepub.com/doi/10.1177/02783649211047890 (visited on 06/19/2023).

[26] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. "ROSPlan: Planning in the Robot Operating System". en. In: *Proceedings of the International Conference on Automated Planning and Scheduling* 25 (Apr. 2015), pp. 333–341. ISSN: 2334-0843. DOI: 10.1609/icaps.v25i1.13699. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/13699 (visited on 06/15/2023).

[27] Nicola Castaman, Enrico Pagello, Emanuele Menegatti, and Alberto Pretto. "Receding Horizon Task and Motion Planning in Changing Environments". en. In: *Robotics and Autonomous Systems* 145 (Nov. 2021), p. 103863. ISSN: 09218890. DOI: 10.1016/j.robot.2021.103863. URL: https://linkinghub.elsevier.com/retrieve/pii/S0921889021001482 (visited on 08/25/2023).

[28] Hendry F. Chame, Ahmadreza Ahmadi, and Jun Tani. "A Hybrid Human-Neurorobotics Approach to Primary Intersubjectivity via Active Inference". In: *Frontiers in Psychology* 11 (Dec. 2020), p. 584869. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2020.584869. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2020.584869/full (visited on 06/16/2023).

[29] Jie Chen and Ron J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Ed. by Kai-Yuan Cai. Vol. 3. The International Series on Asian Studies in Computer and Information Science. Boston, MA: Springer

US, 1999. DOI: 10.1007/978-1-4615-5149-2. URL: http://link.springer.com/10.1007/978-1-4615-5149-2 (visited on 06/19/2023).

[30] Xinjia Chen. "A New Generalization of Chebyshev Inequality for Random Vectors". In: (2007). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.0707.0805. URL: https://arxiv.org/abs/0707.0805 (visited on 06/19/2023).

[31] 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)*. ISSN: 2577-087X. May 2019, pp. 8839–8845. DOI: 10.1109/ICRA.2019.8794128.

[32] Michele Colledanchise and Petter Ögren. "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees". In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017). Conference Name: IEEE Transactions on Robotics, pp. 372–389. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2633567.

[33] Lin Cong, Michael Grner, Philipp Ruppel, Hongzhuo Liang, Norman Hendrich, and Jianwei Zhang. "Self-Adapting Recurrent Models for Object Pushing from Learning in Simulation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 5304–5310. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341076. URL: https://ieeexplore.ieee.org/document/9341076/ (visited on 06/19/2023).

[34] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. "Active inference on discrete state-spaces: A synthesis". en. In: *Journal of Mathematical Psychology* 99 (Dec. 2020), p. 102447. ISSN: 00222496. DOI: 10.1016/j.jmp.2020.102447. (Visited on 06/15/2023).

[35] Lancelot Da Costa, Noor Sajid, Thomas Parr, Karl Friston, and Ryan Smith. *Reward Maximisation through Discrete Active Inference.* arXiv:2009.08111 [cs, math, q-bio]. July 2022. URL: http://arxiv.org/abs/2009.08111 (visited on 06/19/2023).

[36] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "Object Rearrangement Using Learned Implicit Collision Functions". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 6010–6017. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561516. URL: https://ieeexplore.ieee.org/document/9561516/ (visited on 06/19/2023).

[37] W.E. Dixon, I.D. Walker, D.M. Dawson, and J.P. Hartranft. "Fault detection for robot manipulators with parametric uncertainty: a prediction-error-based approach". In: *IEEE Transactions on Robotics and Automation* 16.6 (Dec. 2000), pp. 689–699. ISSN: 1042296X. DOI: 10.1109/70.897780. URL: http://ieeexplore.ieee.org/document/897780/ (visited on 06/19/2023).

[38] Kutluhan Erol, James Hendler, and Dana Nau. "HTN Planning: Complexity and Expressivity". In: 1994.

8

[39] Davide Faconti and Michele Colledanchise. *Behaviortree.cpp*. 2019. URL: https://github.com/BehaviorTree/BehaviorTree.CPP.

[40] Shaoji Fang, Mogens Blanke, and Bernt J. Leira. "Mooring system diagnosis and structural reliability control for position moored vessels". en. In: *Control Engineering Practice* 36 (Mar. 2015), pp. 12–26. ISSN: 09670661. DOI: 10.1016/j.conengprac.2014.11.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0967066114002718 (visited on 06/19/2023).

[41] R. Ferrari, H. Dibowski, and S. Baldi. "A Message Passing Algorithm for Automatic Synthesis of Probabilistic Fault Detectors from Building Automation Ontologies". en. In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 4184–4190. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.809. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896317312636 (visited on 06/19/2023).

[42] Charles W. Fox and Stephen J. Roberts. "A tutorial on variational Bayesian inference". en. In: *Artificial Intelligence Review* 38.2 (Aug. 2012), pp. 85–95. ISSN: 1573-7462. DOI: 10.1007/s10462-011-9236-8. URL: https://doi.org/10.1007/s10462-011-9236-8 (visited on 06/19/2023).

[43] K.J. Friston, N. Trujillo-Barreto, and J. Daunizeau. "DEM: A variational treatment of dynamic systems". en. In: *NeuroImage* 41.3 (July 2008), pp. 849–885. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2008.02.054. URL: https://linkinghub.elsevier.com/retrieve/pii/S1053811908001894 (visited on 06/16/2023).

[44] Karl Friston. "Hierarchical Models in the Brain". en. In: *PLOS Computational Biology* 4.11 (Nov. 2008). Publisher: Public Library of Science, e1000211. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000211. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000211 (visited on 07/24/2023).

[45] Karl Friston. "The free-energy principle: a unified brain theory?" en. In: *Nature Reviews Neuroscience* 11.2 (Feb. 2010). Number: 2 Publisher: Nature Publishing Group, pp. 127–138. ISSN: 1471-0048. DOI: 10.1038/nrn2787. URL: https://www.nature.com/articles/nrn2787 (visited on 06/15/2023).

[46] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O'Doherty, and Giovanni Pezzulo. "Active inference and learning". en. In: *Neuroscience & Biobehavioral Reviews* 68 (Sept. 2016), pp. 862–879. ISSN: 0149-7634. DOI: 10.1016/j.neubiorev.2016.06.022. URL: https://www.sciencedirect.com/science/article/pii/S0149763416301336 (visited on 06/15/2023).

[47] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. "Active Inference: A Process Theory". en. In: *Neural Computation* 29.1 (Jan. 2017), pp. 1–49. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/NECO_a_00912. URL: https://direct.mit.edu/neco/article/29/1/1-49/8207 (visited on 06/15/2023).

8

[48] Karl Friston, Jérémie Mattout, and James Kilner. "Action understanding and active inference". en. In: *Biological Cybernetics* 104.1 (Feb. 2011), pp. 137–160. ISSN: 1432-0770. DOI: 10.1007/s00422-011-0424-z. URL: https://doi.org/10.1007/s00422-011-0424-z (visited on 06/15/2023).

[49] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. "Variational free energy and the Laplace approximation". en. In: *NeuroImage* 34.1 (Jan. 2007), pp. 220–234. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2006.08.035. URL: https://www.sciencedirect.com/science/article/pii/S1053811906008822 (visited on 06/16/2023).

[50] Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas Fitzgerald, and Giovanni Pezzulo. "Active inference and epistemic value". In: *Cognitive Neuroscience* 6.4 (Oct. 2015). Publisher: Routledge _eprint: https://doi.org/10.1080/17588928.2015.1020053, pp. 187–214. ISSN: 1758-8928. DOI: 10.1080/17588928.2015.1020053. URL: https://doi.org/10.1080/17588928.2015.1020053 (visited on 07/24/2023).

[51] Karl Friston, Spyridon Samothrakis, and Read Montague. "Active inference and agency: optimal control without cost functions". en. In: *Biological Cybernetics* 106.8 (Oct. 2012), pp. 523–541. ISSN: 1432-0770. DOI: 10.1007/s00422-012-0512-8. URL: https://doi.org/10.1007/s00422-012-0512-8 (visited on 06/15/2023).

[52] Karl Friston, Klaas Stephan, Baojuan Li, and Jean Daunizeau. "Generalised Filtering". en. In: *Mathematical Problems in Engineering* 2010 (2010), pp. 1–34. ISSN: 1024-123X, 1563-5147. DOI: 10.1155/2010/621670. URL: http://www.hindawi.com/journals/mpe/2010/621670/ (visited on 06/16/2023).

[53] Karl J. Friston, Jean Daunizeau, and Stefan J. Kiebel. "Reinforcement Learning or Active Inference?" en. In: *PLOS ONE* 4.7 (July 2009). Publisher: Public Library of Science, e6421. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0006421. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0006421 (visited on 07/24/2023).

[54] Karl J. Friston, Jean Daunizeau, James Kilner, and Stefan J. Kiebel. "Action and behavior: a free-energy formulation". en. In: *Biological Cybernetics* 102.3 (Mar. 2010), pp. 227–260. ISSN: 1432-0770. DOI: 10.1007/s00422-010-0364-z. URL: https://doi.org/10.1007/s00422-010-0364-z (visited on 06/15/2023).

[55] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrated Task and Motion Planning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (May 2021), pp. 265–293. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-091420-084139. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-091420-084139 (visited on 08/25/2023).

8

[56] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "PDDL-Stream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning". In: *Proceedings of the International Conference on Automated Planning and Scheduling* 30 (June 2020), pp. 440–448. ISSN: 2334-0843, 2334-0835. DOI: 10.1609/icaps.v30i1.6739. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/6739 (visited on 08/25/2023).

[57] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. "Online Replanning in Belief Space for Partially Observable Task and Motion Problems". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2020, pp. 5678–5684. DOI: 10.1109/ICRA40945.2020.9196681.

[58] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. 1st ed. Cambridge University Press, July 2016. ISBN: 978-1-139-58392-3. DOI: 10.1017/CBO9781139583923. URL: https://www.cambridge.org/core/product/identifier/9781139583923/type/book (visited on 06/15/2023).

[59] Malik Ghallab, Dana Nau, and Paolo Traverso. "The actor's view of automated planning and acting: A position paper". en. In: *Artificial Intelligence* 208 (Mar. 2014), pp. 1–17. ISSN: 0004-3702. DOI: 10.1016/j.artint.2013.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0004370213001173 (visited on 06/15/2023).

[60] Jason Harris, Danny Driess, and Marc Toussaint. "FC [3] : Feasibility-Based Control Chain Coordination". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 13769–13776. ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9981758. URL: https://ieeexplore.ieee.org/document/9981758/ (visited on 08/25/2023).

[61] Conor Heins, Beren Millidge, Daphne Demekas, Brennan Klein, Karl Friston, Iain Couzin, and Alexander Tschantz. "pymdp: A Python library for active inference in discrete state spaces". In: *Journal of Open Source Software* 7.73 (May 2022). arXiv:2201.03904 [cs, q-bio], p. 4098. ISSN: 2475-9066. DOI: 10.21105/joss.04098. URL: http://arxiv.org/abs/2201.03904 (visited on 06/15/2023).

[62] Casper Hesp, Ryan Smith, Thomas Parr, Micah Allen, Karl J. Friston, and Maxwell J. D. Ramstead. "Deeply Felt Affect: The Emergence of Valence in Deep Active Inference". en. In: *Neural Computation* 33.2 (Feb. 2021), pp. 398–446. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01341. URL: https://direct.mit.edu/neco/article/33/2/398/95642/Deeply-Felt-Affect-The-Emergence-of-Valence-in (visited on 06/15/2023).

8

[63] Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. "Learning-Based Model Predictive Control: Toward Safe Learning in Control". en. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (May 2020), pp. 269–296. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-090419-075625. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-090419-075625 (visited on 06/19/2023).

[64] Francois R Hogan and Alberto Rodriguez. "Reactive planar non-prehensile manipulation with hybrid model predictive control". en. In: *The International Journal of Robotics Research* 39.7 (June 2020), pp. 755–773. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364920913938. URL: http://journals.sagepub.com/doi/10.1177/0278364920913938 (visited on 06/19/2023).

[65] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. *Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo.* Publisher: arXiv Version Number: 2. 2022. DOI: 10.48550/ARXIV.2212.00541. URL: https://arxiv.org/abs/2212.00541 (visited on 06/19/2023).

[66] Baichuan Huang, Abdeslam Boularias, and Jingjin Yu. "Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Kyoto, Japan: IEEE, Oct. 2022, pp. 1153–1160. ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9981962. URL: https://ieeexplore.ieee.org/document/9981962/ (visited on 08/25/2023).

[67] Orkin Jeff. "Applying Goal-Oriented Action Planning to Games". In: *AI Game Programming Wisdom* 2 (2003), pp. 217–228. URL: https://cir.nii.ac.jp/crid/1573387451004878592 (visited on 06/15/2023).

[68] Jin-Ho Shin and Ju-Jang Lee. "Fault detection and robust fault recovery control for robot manipulators with actuator failures". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation.* Vol. 2. Detroit, MI, USA: IEEE, 1999, pp. 861–866. ISBN: 978-0-7803-5180-6. DOI: 10.1109/ROBOT.1999.772398. URL: http://ieeexplore.ieee.org/document/772398/ (visited on 06/19/2023).

[69] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". en. In: *Artificial Intelligence* 101.1 (May 1998), pp. 99–134. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00023-X. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X (visited on 07/24/2023).

[70] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Hierarchical task and motion planning in the now". In: *2011 IEEE International Conference on Robotics and Automation.* ISSN: 1050-4729. May 2011, pp. 1470–1477. DOI: 10.1109/ICRA.2011.5980391.

8

[71] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Integrated task and motion planning in belief space". en. In: *The International Journal of Robotics Research* 32.9-10 (Aug. 2013). Publisher: SAGE Publications Ltd STM, pp. 1194–1227. ISSN: 0278-3649. DOI: 10.1177/0278364913484072. URL: https://doi.org/10.1177/0278364913484072 (visited on 06/15/2023).

[72] Raphael Kaplan and Karl J. Friston. "Planning and navigation as active inference". en. In: *Biological Cybernetics* 112.4 (Aug. 2018), pp. 323–343. ISSN: 1432-0770. DOI: 10.1007/s00422-018-0753-2. URL: https://doi.org/10.1007/s00422-018-0753-2 (visited on 06/15/2023).

[73] Hilbert J. Kappen. "Linear Theory for Control of Nonlinear Stochastic Systems". en. In: *Physical Review Letters* 95.20 (Nov. 2005), p. 200201. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.95.200201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.95.200201 (visited on 08/14/2023).

[74] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". en. In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911406761. URL: http://journals.sagepub.com/doi/10.1177/0278364911406761 (visited on 06/19/2023).

[75] J. Krüger, T.K. Lien, and A. Verl. "Cooperation of human and machines in assembly lines". en. In: *CIRP Annals* 58.2 (2009), pp. 628–646. ISSN: 00078506. DOI: 10.1016/j.cirp.2009.09.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0007850609001760 (visited on 06/19/2023).

[76] Thijs van de Laar, Ayça Özçelikkale, and Henk Wymeersch. "Application of the Free Energy Principle to Estimation and Control". In: *IEEE Transactions on Signal Processing* 69 (2021). Conference Name: IEEE Transactions on Signal Processing, pp. 4234–4244. ISSN: 1941-0476. DOI: 10.1109/TSP.2021.3095711.

[77] Pablo Lanillos and Gordon Cheng. "Adaptive Robot Body Learning and Estimation Through Predictive Coding". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2018, pp. 4083–4090. DOI: 10.1109/IROS.2018.8593684.

[78] Pablo Lanillos, Cristian Meo, Corrado Pezzato, Ajith Anil Meera, Mohamed Baioumy, Wataru Ohata, Alexander Tschantz, Beren Millidge, Martijn Wisse, Christopher L. Buckley, and Jun Tani. *Active Inference in Robotics and Artificial Agents: Survey and Challenges*. en. Dec. 2021. URL: https://arxiv.org/abs/2112.01871v1 (visited on 06/15/2023).

[79] Teguh Santoso Lembono, Antonio Paolillo, Emmanuel Pignat, and Sylvain Calinon. "Memory of Motion for Warm-Starting Trajectory Optimization". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 2594–2601. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2972893. (Visited on 09/27/2023).

8

[80] Martin Levihn, Leslie Pack Kaelbling, Tomas Lozano-Pérez, and Mike Stilman. "Foresight and reconsideration in hierarchical planning and execution". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Nov. 2013, pp. 224–231. DOI: 10.1109/IROS.2013.6696357.

[81] Anqi Li, Mustafa Mukadam, Magnus Egerstedt, and Byron Boots. "Multi-objective Policy Generation for Multi-robot Systems Using Riemannian Motion Policies". en. In: *Robotics Research*. Ed. by Tamim Asfour, Eiichi Yoshida, Jaeheung Park, Henrik Christensen, and Oussama Khatib. Springer Proceedings in Advanced Robotics. Cham: Springer International Publishing, 2022, pp. 258–274. ISBN: 978-3-030-95459-8. DOI: 10.1007/978-3-030-95459-8_16.

[82] Shen Li, Daehyung Park, Yoonchang Sung, Julie A. Shah, and Nicholas Roy. "Reactive Task and Motion Planning under Temporal Logic Specifications". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 12618–12624. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561807. URL: https://ieeexplore.ieee.org/document/9561807/ (visited on 08/25/2023).

[83] D. V. Lindley. *Bayesian Statistics: A Review*. en. Society for Industrial and Applied Mathematics, Jan. 1972. DOI: 10.1137/1.9781611970654. URL: http://epubs.siam.org/doi/book/10.1137/1.9781611970654 (visited on 06/19/2023).

[84] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2020, pp. 2718–2725. DOI: 10.1109/IROS45743.2020.9341207.

[85] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. Curran, 2021.

[86] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse. "Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane: IEEE, May 2018, pp. 2986–2993. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8463154. (Visited on 09/27/2023).

[87] Francisco Martín Rico, Matteo Morelli, Huascar Espinoza, Francisco Rodríguez-Lera, and Vicente Matellán Olivera. "Optimized Execution of PDDL Plans using Behavior Trees". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '21. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2021, pp. 1596–1598. ISBN: 978-1-4503-8307-3. (Visited on 06/19/2023).

8

[88] Ajith Anil Meera and Martijn Wisse. "Free Energy Principle Based State and Input Observer Design for Linear Systems with Colored Noise". In: *2020 American Control Conference (ACC)*. ISSN: 2378-5861. July 2020, pp. 5052–5058. DOI: 10.23919/ACC45564.2020.9147581.

[89] Cristian Meo and Pablo Lanillos. "Multimodal VAE Active Inference Controller". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Sept. 2021, pp. 2693–2699. DOI: 10.1109/IROS51168.2021.9636394.

[90] Toki Migimatsu and Jeannette Bohg. "Object-Centric Task and Motion Planning in Dynamic Environments". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 844–851. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2965875. URL: https://ieeexplore.ieee.org/document/8957055/ (visited on 08/25/2023).

[91] Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. "Whence the Expected Free Energy?" en. In: *Neural Computation* 33.2 (Feb. 2021), pp. 447–482. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01354. URL: https://direct.mit.edu/neco/article/33/2/447/95645/Whence-the-Expected-Free-Energy (visited on 07/24/2023).

[92] Joao Moura, Theodoros Stouraitis, and Sethu Vijayakumar. "Non-prehensile Planar Manipulation via Trajectory Optimization with Complementarity Constraints". In: *International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022, pp. 970–976. ISBN: 978-1-72819-681-7. DOI: 10.1109/ICRA46639.2022.9811942. URL: https://ieeexplore.ieee.org/document/9811942/ (visited on 06/19/2023).

[93] Esmaeil Najafi, Robert Babuška, and Gabriel A.D. Lopes. "An application of sequential composition control to cooperative systems". In: *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*. July 2015, pp. 15–20. DOI: 10.1109/RoMoCo.2015.7219707.

[94] K.S. Narendra and J. Balakrishnan. "Adaptive control using multiple models". In: *IEEE Transactions on Automatic Control* 42.2 (Feb. 1997), pp. 171–187. ISSN: 00189286. DOI: 10.1109/9.554398. URL: http://ieeexplore.ieee.org/document/554398/ (visited on 06/19/2023).

[95] Dana Nau, Malik Ghallab, and Paolo Traverso. "Blended Planning and Acting: Preliminary Approach, Research Challenges". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Mar. 2015). Number: 1. ISSN: 2374-3468. DOI: 10.1609/aaai.v29i1.9768. URL: https://ojs.aaai.org/index.php/AAAI/article/view/9768 (visited on 06/15/2023).

[96] Petter Ögren and Michele Colledanchise. *Behavior Trees in Robotics and AI: An Introduction*. eng. CRC Press, 2018. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-232942 (visited on 06/15/2023).

8

[97] Wataru Ohata and Jun Tani. "Investigation of the Sense of Agency in Social Cognition, Based on Frameworks of Predictive Coding and Active Inference: A Simulation Study on Multimodal Imitative Interaction". In: *Frontiers in Neurorobotics* 14 (Sept. 2020), p. 61. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.00061. URL: https://www.frontiersin.org/article/10.3389/fnbot.2020.00061/full (visited on 06/16/2023).

[98] Guillermo Oliver, Pablo Lanillos, and Gordon Cheng. "An Empirical Study of Active Inference on a Humanoid Robot". In: *IEEE Transactions on Cognitive and Developmental Systems* 14.2 (June 2022). Conference Name: IEEE Transactions on Cognitive and Developmental Systems, pp. 462–471. ISSN: 2379-8939. DOI: 10.1109/TCDS.2021.3049907.

[99] Laurentz E. Olivier and Ian K. Craig. "Fault-tolerant nonlinear mpc using particle filtering". In: *IFAC-PapersOnLine* 49.7 (2016). Publisher: Elsevier, pp. 177–182.

[100] Jeff Orkin. "Three States and a Plan: The A.I. of F.E.A.R." en. In: (2006).

[101] Joaquim Ortiz-Haro, Erez Karpas, Michael Katz, and Marc Toussaint. "A Conflict-Driven Interface Between Symbolic Planning and Nonlinear Constraint Solving". In: *IEEE Robotics and Automation Letters* 7.4 (Oct. 2022), pp. 10518–10525. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3191948. URL: https://ieeexplore.ieee.org/document/9832718/ (visited on 08/25/2023).

[102] Pierre-Yves Oudeyer and Frederic Kaplan. "What is intrinsic motivation? A typology of computational approaches". In: *Frontiers in Neurorobotics* 1 (2007). ISSN: 1662-5218. URL: https://www.frontiersin.org/articles/10.3389/neuro.12.006.2007 (visited on 07/24/2023).

[103] Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-draa. "An online POMDP algorithm for complex multiagent environments". In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. AAMAS '05. New York, NY, USA: Association for Computing Machinery, July 2005, pp. 970–977. ISBN: 978-1-59593-093-4. DOI: 10.1145/1082473.1082620. URL: https://doi.org/10.1145/1082473.1082620 (visited on 06/15/2023).

[104] Thomas Parr and Karl J. Friston. "Generalised free energy and active inference". en. In: *Biological Cybernetics* 113.5 (Dec. 2019), pp. 495–513. ISSN: 1432-0770. DOI: 10.1007/s00422-019-00805-w. URL: https://doi.org/10.1007/s00422-019-00805-w (visited on 07/24/2023).

[105] Gaetano Paviglianiti, Francesco Pierri, Fabrizio Caccavale, and Massimiliano Mattei. "Robust fault detection and isolation for proprioceptive sensors of robot manipulators". en. In: *Mechatronics* 20.1 (Feb. 2010), pp. 162–170. ISSN: 09574158. DOI: 10.1016/j.mechatronics.2009.09.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957415809001548 (visited on 06/19/2023).

8

[106] Chris Paxton, Nathan Ratliff, Clemens Eppner, and Dieter Fox. "Representing Robot Task Plans as Robust Logical-Dynamical Systems". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. 2019, pp. 5588–5595. DOI: 10.1109/IROS40897.2019.8967861.

[107] Corrado Pezzato, Mohamed Baioumy, Carlos Hernandez Corbato, Nick Hawes, Martijn Wisse, and Riccardo Ferrari. "Active inference for fault tolerant control of robot manipulators with sensory faults". In: *Active Inference: First International Workshop, IWAI 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14, 2020, Proceedings 1*. Springer, 2020, pp. 20–27.

[108] Corrado Pezzato, Carlos Hernández Corbato, Stefan Bonhof, and Martijn Wisse. "Active Inference and Behavior Trees for Reactive Action Planning and Execution in Robotics". In: *IEEE Transactions on Robotics* 39.2 (Apr. 2023). Conference Name: IEEE Transactions on Robotics, pp. 1050–1069. ISSN: 1941-0468. DOI: 10.1109/TRO.2022.3226144.

[109] 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 (Apr. 2020). Conference Name: IEEE Robotics and Automation Letters, pp. 2973–2980. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.2974451.

[110] Corrado Pezzato, Chadi Salmi, Elia Trevisan, Javier Alonso Mora, and Carlos Hernandez Corbato. "Sampling-Based MPC Using a GPU-parallelizable Physics Simulator as Dynamic Model: an Open Source Implementation with IsaacGym". en. In: pezzato_sampling_ws_2023. May 2023. URL: https://openreview.net/forum?id=fvfZKL1hCx (visited on 06/15/2023).

[111] Tim Pfeifer, Sven Lange, and Peter Protzel. "Dynamic Covariance Estimation — A parameter free approach to robust Sensor Fusion". In: *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Nov. 2017, pp. 359–365. DOI: 10.1109/MFI.2017.8170347.

[112] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press, 2006. ISBN: 978-0-262-18253-9.

[113] Nathan D. Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Muhammad Asif Rana. "Generalized Nonlinear and Finsler Geometry for Robotics". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 10206–10212. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561543. URL: https://ieeexplore.ieee.org/document/9561543/ (visited on 06/19/2023).

[114] Vahab Rostampour, Riccardo Ferrari, and Tamas Keviczky. "A set based probabilistic approach to threshold design for optimal fault detection". In: *2017 American Control Conference (ACC)*. Seattle, WA, USA: IEEE, May 2017, pp. 5422–5429. ISBN: 978-1-5090-5992-8. DOI: 10.23919/ACC.2017.

8

7963798. URL: https://ieeexplore.ieee.org/document/7963798/ (visited on 06/19/2023).

[115] Vahab Rostampour, Riccardo Ferrari, André M.H. Teixeira, and Tamás Keviczky. "Differentially-Private Distributed Fault Diagnosis for Large-Scale Nonlinear Uncertain Systems". en. In: *IFAC-PapersOnLine* 51.24 (2018), pp. 975–982. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.09.703. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896318324236 (visited on 06/19/2023).

[116] Vahab Rostampour, Riccardo M.G. Ferrari, Andre M.H. Teixeira, and Tamas Keviczky. "Privatized Distributed Anomaly Detection for Large-Scale Nonlinear Uncertain Systems". In: *IEEE Transactions on Automatic Control* 66.11 (Nov. 2021), pp. 5299–5313. ISSN: 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2020.3040251. URL: https://ieeexplore.ieee.org/document/9268470/ (visited on 06/19/2023).

[117] Evgenii Safronov, Michele Colledanchise, and Lorenzo Natale. "Task Planning with Belief Behavior Trees". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2020, pp. 6870–6877. DOI: 10.1109/IROS45743.2020.9341562.

[118] Noor Sajid, Philip J. Ball, Thomas Parr, and Karl J. Friston. "Active Inference: Demystified and Compared". en. In: *Neural Computation* 33.3 (Mar. 2021), pp. 674–712. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01357. URL: https://direct.mit.edu/neco/article/33/3/674-712/97486 (visited on 06/15/2023).

[119] Jürgen Schmidhuber. "Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010)". In: *IEEE Transactions on Autonomous Mental Development* 2.3 (Sept. 2010). Conference Name: IEEE Transactions on Autonomous Mental Development, pp. 230–247. ISSN: 1943-0612. DOI: 10.1109/TAMD.2010.2056368.

[120] Philipp Schwartenbeck, Johannes Passecker, Tobias U Hauser, Thomas HB FitzGerald, Martin Kronbichler, and Karl J Friston. "Computational mechanisms of curiosity and goal-directed exploration". In: *eLife* 8 (May 2019). Ed. by Michael J Frank. Publisher: eLife Sciences Publications, Ltd, e41703. ISSN: 2050-084X. DOI: 10.7554/eLife.41703. URL: https://doi.org/10.7554/eLife.41703 (visited on 06/15/2023).

[121] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. "Review on model predictive control: an engineering perspective". en. In: *The International Journal of Advanced Manufacturing Technology* 117.5 (Nov. 2021), pp. 1327–1349. ISSN: 1433-3015. DOI: 10.1007/s00170-021-07682-3. URL: https://doi.org/10.1007/s00170-021-07682-3 (visited on 06/19/2023).

[122] Sarah Schwöbel, Stefan Kiebel, and Dimitrije Marković. "Active Inference, Belief Propagation, and the Bethe Approximation". en. In: *Neural Computation* 30.9 (Sept. 2018), pp. 2530–2567. ISSN: 0899-7667, 1530-888X. DOI:

8

10.1162/neco_a_01108. URL: https://direct.mit.edu/neco/article/30/9/2530-2567/8396 (visited on 06/15/2023).

[123] Nicola Scianca, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. "MPC for Humanoid Gait Generation: Stability and Feasibility". In: *IEEE Transactions on Robotics* 36.4 (Aug. 2020), pp. 1171–1188. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2019.2958483. URL: https://ieeexplore.ieee.org/document/8955951/ (visited on 08/25/2023).

[124] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics. Modelling, Planning and Control.* Advanced Textbooks in Control and Signal Processing. London: Springer London, 2009. DOI: 10.1007/978-1-84628-642-1. URL: http://link.springer.com/10.1007/978-1-84628-642-1 (visited on 06/19/2023).

[125] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots.* 2nd ed. Intelligent robotics and autonomous agents. OCLC: ocn649700153. Cambridge, Mass: MIT Press, 2011. ISBN: 978-0-262-01535-6.

[126] Ryan Smith, Karl J. Friston, and Christopher J. Whyte. "A step-by-step tutorial on active inference and its application to empirical data". en. In: *Journal of Mathematical Psychology* 107 (Apr. 2022), p. 102632. ISSN: 0022-2496. DOI: 10.1016/j.jmp.2021.102632. URL: https://www.sciencedirect.com/science/article/pii/S0022249621000973 (visited on 06/15/2023).

[127] Mark Solms. *The hidden spring: a journey to the source of consciousness.* eng. London: Profile Books, 2021. ISBN: 978-1-78816-283-8.

[128] 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).* Xi'an, China: IEEE, May 2021, pp. 12759–12765. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561821. URL: https://ieeexplore.ieee.org/document/9561821/ (visited on 08/25/2023).

[129] Max Spahn, Chadi Salmi, and Javier Alonso-Mora. *Local Planner Bench: Benchmarking for Local Motion Planning.* Publisher: arXiv Version Number: 1. 2022. DOI: 10.48550/ARXIV.2210.06033. URL: https://arxiv.org/abs/2210.06033 (visited on 06/19/2023).

[130] Max Spahn, Martijn Wisse, and Javier Alonso-Mora. "Dynamic Optimization Fabrics for Motion Generation". In: *IEEE Transactions on Robotics* (2023), pp. 1–16. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2023.3255587. URL: https://ieeexplore.ieee.org/document/10086617/ (visited on 06/19/2023).

[131] Evangelos Theodorou. "Nonlinear Stochastic Control and Information Theoretic Dualities: Connections, Interdependencies and Thermodynamic Interpretations". en. In: *Entropy* 17.5 (May 2015), pp. 3352–3375. ISSN: 1099-4300. DOI: 10.3390/e17053352. URL: http://www.mdpi.com/1099-4300/17/5/3352 (visited on 08/14/2023).

[132] Evangelos A. Theodorou and Emanuel Todorov. "Relative entropy and free energy dualities: Connections to Path Integral and KL control". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. Maui, HI, USA: IEEE, Dec. 2012, pp. 1466–1473. ISBN: 978-1-4673-2066-5. DOI: 10.1109/CDC.2012.6426381. URL: http://ieeexplore.ieee.org/document/6426381/ (visited on 08/14/2023).

[133] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109. URL: http://ieeexplore.ieee.org/document/6386109/ (visited on 06/19/2023).

[134] Marc Toussaint, Jason Harris, Jung-Su Ha, Danny Driess, and Wolfgang Honig. "Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 13753–13760. ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9982236. URL: https://ieeexplore.ieee.org/document/9982236/ (visited on 08/25/2023).

[135] Alexander Tschantz, Beren Millidge, Anil K. Seth, and Christopher L. Buckley. *Reinforcement Learning through Active Inference*. arXiv:2002.12636 [cs, eess, math, stat]. Feb. 2020. DOI: 10.48550/arXiv.2002.12636. URL: http://arxiv.org/abs/2002.12636 (visited on 06/16/2023).

[136] Mien Van, Denglu Wu, Shuzhi Sam Ge, and Hongliang Ren. "Fault Diagnosis in Image-Based Visual Servoing With Eye-in-Hand Configurations Using Kalman Filter". In: *IEEE Transactions on Industrial Informatics* 12.6 (Dec. 2016), pp. 1998–2007. ISSN: 1551-3203, 1941-0050. DOI: 10.1109/TII.2016.2590338. URL: http://ieeexplore.ieee.org/document/7508912/ (visited on 06/19/2023).

[137] Thijs Van De Laar, İsmail Şenöz, Ayça Özçelikkale, and Henk Wymeersch. "Chance-Constrained Active Inference". en. In: *Neural Computation* 33.10 (Sept. 2021), pp. 2710–2735. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01427. URL: https://direct.mit.edu/neco/article/33/10/2710/103013/Chance-Constrained-Active-Inference (visited on 06/16/2023).

[138] William Vega-Brown and Nicholas Roy. "CELLO-EM: Adaptive sensor models without ground truth". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Nov. 2013, pp. 1907–1914. DOI: 10.1109/IROS.2013.6696609.

[139] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. "Robotic fault detection and fault tolerance: A survey". en. In: *Reliability Engineering & System Safety* 46.2 (Jan. 1994), pp. 139–158. ISSN: 09518320. DOI: 10.1016/0951-8320(94)90132-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0951832094901325 (visited on 06/19/2023).

8

[140] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. "Model Predictive Path Integral Control: From Theory to Parallel Computation". en. In: *Journal of Guidance, Control, and Dynamics* 40.2 (Feb. 2017), pp. 344–357. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G001921. URL: https://arc.aiaa.org/doi/10.2514/1.G001921 (visited on 06/19/2023).

[141] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. "Aggressive driving with model predictive path integral control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, May 2016, pp. 1433–1440. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487277. URL: https://ieeexplore.ieee.org/document/7487277/ (visited on 08/14/2023).

[142] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving". In: *IEEE Transactions on Robotics* 34.6 (Dec. 2018), pp. 1603–1622. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2018.2865891. URL: https://ieeexplore.ieee.org/document/8558663/ (visited on 06/19/2023).

[143] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. "Information theoretic MPC for model-based reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore: IEEE, May 2017, pp. 1714–1721. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989202. URL: https://ieeexplore.ieee.org/document/7989202/ (visited on 08/25/2023).

[144] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. "DESPOT: Online POMDP Planning with Regularization". In: *Journal of Artificial Intelligence Research* 58 (Jan. 2017), pp. 231–266. ISSN: 1076-9757. DOI: 10.1613/jair.5328. URL: https://jair.org/index.php/jair/article/view/11043 (visited on 06/15/2023).

[145] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs". en. In: *International Journal of Control* 93.1 (Jan. 2020), pp. 13–29. ISSN: 0020-7179, 1366-5820. DOI: 10.1080/00207179.2017.1316017. URL: https://www.tandfonline.com/doi/full/10.1080/00207179.2017.1316017 (visited on 06/19/2023).

[146] Youmin Zhang and Jin Jiang. "Bibliographical review on reconfigurable fault-tolerant control systems". en. In: *Annual Reviews in Control* 32.2 (Dec. 2008), pp. 229–252. ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2008.03.008. (Visited on 07/25/2023).

[147] Yuezhe Zhang, Corrado Pezzato, Elia Trevisan, Chadi Salmi, Carlos Hernandez Corbato, and Javier Alonso-Mora. *Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning*. arXiv:2312.02328 [cs]. Dec. 2023. URL: https://arxiv.org/abs/2312.02328 (visited on 12/04/2023).

8

[148]   Ziyi Zhou, Dong Jae Lee, Yuki Yoshinaga, Stephen Balakirsky, Dejun Guo, and Ye Zhao. "Reactive Task Allocation and Planning for Quadrupedal and Wheeled Robot Teaming". In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. Mexico City, Mexico: IEEE, Aug. 2022, pp. 2110–2117. ISBN: 978-1-66549-042-9. DOI: 10.1109/CASE49997.2022.9926658. URL: https://ieeexplore.ieee.org/document/9926658/ (visited on 08/25/2023).

A

Appendix

A.1. Generative models

Consider the generative model in active inference $P(\bar{o}, \bar{s}, \boldsymbol{\eta}, \pi)$. By using the chain rule, we can write:

$$P(\bar{o}, \bar{s}, \boldsymbol{\eta}, \pi) = P(\bar{o}|\bar{s}, \boldsymbol{\eta}, \pi)P(\bar{s}|\boldsymbol{\eta}, \pi)P(\boldsymbol{\eta}|\pi)P(\pi) \tag{A.1}$$

Note that $\bar{o}$ is conditionally independent from the model parameters $\boldsymbol{\eta}$ and π given $\bar{s}$. In addition, under the Markov property, the next state and current observations depend only on the current state:

$$P(\bar{o}|\bar{s}, \boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(o_\tau|s_\tau) \tag{A.2}$$

The model is further simplified considering that $\bar{s}$ and $\boldsymbol{\eta}$ are conditionally independent given π:

$$P(\bar{s}|\boldsymbol{\eta}, \pi) = \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi) \tag{A.3}$$

Finally, consider the model parameters explicitly:

$$P(\bar{o}, \bar{s}, \boldsymbol{\eta}, \pi) = P(\bar{o}, \bar{s}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \pi) =$$

$$P(\pi)P(\boldsymbol{A})P(\boldsymbol{B})P(\boldsymbol{D}) \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi)P(o_\tau|s_\tau) \tag{A.4}$$

$P(\boldsymbol{A}), P(\boldsymbol{B}), P(\boldsymbol{D})$ are Dirichlet distributions over the model parameters [47]. In case the model parameters are fixed by the user, as in this work, it holds:

$$P(\bar{o}, \bar{s}, \pi) = P(\pi) \prod_{\tau=1}^{T} P(s_\tau|s_{\tau-1}, \pi)P(o_\tau|s_\tau) \tag{A.5}$$

Given the generative model above, we are interested in finding the posterior hidden causes of sensory data. For the sake of these derivations, we consider that the parameters associated with the task are known and do not introduce uncertainty. Using Bayes rule:

$$P(\bar{s}, \pi | \bar{o}) = \frac{P(\bar{o} | \bar{s}, \pi) P(\bar{s}, \pi)}{P(\bar{o})} \tag{A.6}$$

Computing the model evidence $P(\bar{o})$ exactly is a well-known and often intractable problem in Bayesian statistics. The exact posterior is then computed minimizing the Kullback-Leibler divergence (D_{KL}, or KL-Divergence) with respect to an approximate posterior distribution $Q(\bar{s}, \pi)$. Doing so, we can define the free-energy as a functional of approximate posterior beliefs which result in an upper bound on surprise. By definition D_{KL} is a non-negative quantity given by the expectation of the logarithmic difference between $Q(\bar{s}, \pi)$ and $P(\bar{s}, \pi | \bar{o})$. Applying the KL-Divergence:

$$\begin{aligned} D_{KL}\left[Q(\bar{s}, \pi) || P(\bar{s}, \pi | \bar{o})\right] = \\ \mathbb{E}_{Q(\bar{s}, \pi)}\left[\ln Q(\bar{s}, \pi) - \ln P(\bar{s}, \pi | \bar{o})\right] \geq 0 \end{aligned} \tag{A.7}$$

D_{KL} is the information loss when Q is used instead of P. Considering equation (A.6) and the chain rule, equation (A.7) can be rewritten as:

$$\begin{aligned} D_{KL}\left[\cdot\right] &= \mathbb{E}_{Q(\bar{s}, \pi)}\left[\ln Q(\bar{s}, \pi) - \ln \frac{P(\bar{o}, \bar{s}, \pi)}{P(\bar{o})}\right] \\ &= \underbrace{\mathbb{E}_{Q(\bar{s}, \pi)}\left[\ln Q(\bar{s}, \pi) - \ln P(\bar{o}, \bar{s}, \pi)\right]}_{F[Q(\bar{s}, \pi)]} + \ln P(\bar{o}) \end{aligned} \tag{A.8}$$

We have just defined the free-energy as the upper bound of surprise:

$$F\left[Q(\bar{s}, \pi)\right] \geq -\ln P(\bar{o}) \tag{A.9}$$

A.2. Variational Free-energy

To fully characterize the free-energy in equation (A.8), we need to specify a form for the approximate posterior $Q(\bar{s}, \pi)$. There are different ways to choose a family of probability distributions [122], compromising between complexity and accuracy of the approximation. In this work, we choose the mean-field approximation. It holds:

$$Q(\bar{s}, \pi) = Q(\bar{s} | \pi) Q(\pi) = Q(\pi) \prod_{\tau=1}^{T} Q(s_\tau | \pi) \tag{A.10}$$

Under mean-field approximation, the plan-dependent states at each time step are approximately independent of the states at any other time step. We can now find an expression for the variational free-energy. Considering the mean-field approximation

and the generative model in eq. (A.5) we can write:

$$F\left[Q(\bar{s}, \pi)\right] = \mathbb{E}_{Q(\bar{s}, \pi)}\left[\ln Q(\pi) + \sum_{\tau=1}^{T} \ln Q(s_\tau | \pi)\right.$$

$$\left. - \ln P(\pi) - \sum_{\tau=1}^{T} \ln P(s_\tau | s_{\tau-1}, \pi) - \sum_{\tau=1}^{T} \ln P(o_\tau | s_\tau)\right] \tag{A.11}$$

Since $Q(\bar{s}, \pi) = Q(\bar{s}|\pi)Q(\pi)$, and since the expectation of a sum is the sum of the expectation, we can write:

$$F\left[\cdot\right] = D_{KL}\left[Q(\pi)||P(\pi)\right] + \mathbb{E}_{Q(\pi)}\left[F(\pi)\left[Q(\bar{s}|\pi)\right]\right] \tag{A.12}$$

where

$$F(\pi)\left[Q(\bar{s}|\pi)\right] = \mathbb{E}_{Q(\bar{s}|\pi)}\left[\sum_{\tau=1}^{T} \ln Q(s_\tau | \pi)\right.$$

$$\left. - \sum_{\tau=1}^{T} \ln P(s_\tau | s_{t-\tau}, \pi) - \sum_{\tau=1}^{T} \ln P(o_\tau | s_\tau)\right] \tag{A.13}$$

One can notice that $F(\pi)$ is accumulated over time, or in other words, it is the sum of free energies over time and plans:

$$F(\pi) = \sum_{\tau=1}^{T} F(\pi, \tau) \tag{A.14}$$

Substituting the agent's belief about the current state at time τ given π with $\boldsymbol{s}_\tau^\pi$, we obtain a matrix form for $F(\pi, \tau)$ that we can compute given the generative model:

$$F(\pi) = \sum_{\tau=1}^{T} \boldsymbol{s}_\tau^{\pi\top}\left[\ln \boldsymbol{s}_\tau^\pi - \ln\left(\boldsymbol{B}_{a_{\tau-1}}\boldsymbol{s}_{\tau-1}^\pi\right) - \ln\left(\boldsymbol{A}^\top \boldsymbol{o}_\tau\right)\right] \tag{A.15}$$

Given a plan π, the probability of state transition $P(s_\tau | s_{\tau-1}, \pi)$ is given by the transition matrix under plan π at time τ, multiplied by the probability of the state at the previous time step. In the special case of $\tau = 1$, we can write:

$$F(\pi, 1) = \boldsymbol{s}_1^{\pi\top}\left[\ln \boldsymbol{s}_1^\pi - \ln \boldsymbol{D} - \ln\left(\boldsymbol{A}^\top \boldsymbol{o}_1\right)\right] \tag{A.16}$$

Finally, we can compute the expectation of the plan-dependent variational free-energy $F(\pi)$ as $\mathbb{E}_{Q(\pi)}\left[F(\pi)\right] = \boldsymbol{\pi}^\top \boldsymbol{F}_\pi$. We indicate $\boldsymbol{F}_\pi = (F(\pi_1), F(\pi_2)...)^\top$ for every allowable plan. To derive state and plan updates that minimize free-energy, F in equation (A.12) is partially differentiated and set to zero, as we will see in the next appendixes.

A.3. State estimation

We differentiate F with respect to the sufficient statistics of the probability distribution of the states. Note that the only part of F dependent on the states is $F(\pi)$. Then:

$$\frac{\partial F}{\partial s_\tau^\pi} = \frac{\partial F}{\partial F(\pi)} \frac{\partial F(\pi)}{\partial s_\tau^\pi} = \boldsymbol{\pi}^\top \big[\mathbf{1} + \ln s_\tau^\pi - \ln\left(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi\right)$$
$$- \ln\left(\boldsymbol{B}_{a_\tau}^\top \boldsymbol{s}_{\tau+1}^\pi\right) - \ln\left(\boldsymbol{A}^\top \boldsymbol{o}_\tau\right)\big] \tag{A.17}$$

Setting the gradient to zero and using the softmax function for normalization:

$$\boldsymbol{s}_\tau^\pi = \sigma(\ln\left(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi\right) + \ln\left(\boldsymbol{B}_{a_\tau}^\top \boldsymbol{s}_{\tau+1}^\pi\right) + \ln\left(\boldsymbol{A}^\top \boldsymbol{o}_\tau\right)) \tag{A.18}$$

Note that the softmax function is insensitive to the constant $\mathbf{1}$. Also, for $\tau = 1$ the term $\ln\left(\boldsymbol{B}_{a_{\tau-1}} \boldsymbol{s}_{\tau-1}^\pi\right)$ is replaced by $\boldsymbol{D}$. Finally, $\ln\left(\boldsymbol{A}^\top \boldsymbol{o}_\tau\right)$ contributes only to past and present time steps, so for this term is null for $t < \tau \leq T$ since those observations are still to be received.

A.4. Expected Free-energy

We indicate with $G(\pi)$ the expected free-energy obtained over future time steps until the time horizon T while following a plan π. Basically, this is the variational free-energy of future trajectories which measures the plausibility of plans according to future predicted observations [118]. To compute it we take the expectation of variational free-energy under the posterior predictive distribution $P(o_\tau|s_\tau)$. Following Sajid et al. [118] we can write:

$$G(\pi) = \sum_{\tau=t+1}^{T} G(\pi, \tau) \tag{A.19}$$

then:

$$G(\pi, \tau) = \mathbb{E}_{\tilde{Q}}\big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau|s_{\tau-1})\big]$$
$$= \mathbb{E}_{\tilde{Q}}\big[\ln Q(s_\tau|\pi) - \ln P(s_\tau|o_\tau, s_{\tau-1}) - \ln P(o_\tau)\big] \tag{A.20}$$

where $\tilde{Q} = P(o_\tau|s_\tau)Q(s_\tau|\pi)$. The expected free-energy is:

$$G(\pi, \tau) \geq \mathbb{E}_{\tilde{Q}}\big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau, s_{\tau-1}, \pi) - \ln P(o_\tau)\big] \tag{A.21}$$

Equivalently, we can express the expected free-energy in terms of preferred observations [34]:

$$G(\pi, \tau) = \mathbb{E}_{\tilde{Q}}\big[\ln Q(o_\tau|\pi) - \ln Q(o_\tau|s_\tau, s_{\tau-1}, \pi) - \ln P(o_\tau)\big] \tag{A.22}$$

Making use of $Q(o_\tau|s_\tau, \pi) = P(o_\tau|s_\tau)$ since the predicted observations in the future are only based on $\boldsymbol{A}$ which is plan independent given s_τ, we have:

$$G(\pi, \tau) = \underbrace{D_{KL}\left[Q(o_\tau|\pi)||P(o_\tau)\right]}_{Expected\ cost} + \underbrace{\mathbb{E}_{Q(s_\tau|\pi)}\left[H(P(o_\tau|s_\tau))\right]}_{Entropy} \tag{A.23}$$

were $H[P(o_\tau|s_\tau)] = \mathbb{E}_{P(o_\tau|s_\tau)}[-\ln P(o_\tau|s_\tau)]$ is the entropy. We are now ready to express the expected free-energy in matrix form, such that we can compute it. From the previous equation, one can notice that plan selection aims at minimizing the expected cost and ambiguity. The latter relates to the uncertainty about future observations given hidden states. In a sense, plans tend to bring the agent to future states that generate unambiguous information over states. On the other hand, the cost is the difference between predicted and prior beliefs about final states. Plans are more likely if they minimize cost, and lead to observations that match prior desires. Minimizing G leads to both exploitative (cost minimizing) and explorative (ambiguity minimizing) behavior. This results in a balance between goal-oriented and novelty-seeking behaviors

Substituting the sufficient statistics in equation (A.23), and recalling that the generative model specifies $P(o_\tau) = \boldsymbol{C}$, one obtains [126]:

$$G(\pi, \tau) = \underbrace{\boldsymbol{o}_\tau^{\pi\top}[\ln \boldsymbol{o}_\tau^\pi - \ln \boldsymbol{C}]}_{Reward\ seeking} \underbrace{-diag(\boldsymbol{A}^\top \ln \boldsymbol{A})^\top \boldsymbol{s}_\tau^\pi}_{Information\ seeking} \tag{A.24}$$

Note that prior preferences are passed through the softmax function before computing the logarithm.

A.5. Updating plan distribution

The update rule for the distribution over possible plans follows directly from the variational free-energy:

$$F[\cdot] = D_{KL}[Q(\pi)||P(\pi)] + \boldsymbol{\pi}^\top \boldsymbol{F}_\pi \tag{A.25}$$

The first term of the equation above can be written as:

$$D_{KL}[Q(\pi)||P(\pi)] = \mathbb{E}_{Q(\pi)}[\ln Q(\pi) - \ln P(\pi)] \tag{A.26}$$

Recalling that the approximate posterior over policies is a softmax function of the expected free-energy $Q(\pi) = \sigma(-G(\pi))$ [34, 47], and taking the gradient with respect to $\boldsymbol{\pi}$ it results:

$$\frac{\partial F}{\partial \boldsymbol{\pi}} = \ln \boldsymbol{\pi} + \boldsymbol{G}_\pi + \boldsymbol{F}_\pi + \boldsymbol{1} \tag{A.27}$$

where $\boldsymbol{G}_\pi = (G(\pi_1), G(\pi_2), ...)^\top$ Finally, setting the gradient to zero and normalizing through softmax, the posterior distribution over plans is obtained:

$$\boldsymbol{\pi} = \sigma(-\boldsymbol{G}_\pi - \boldsymbol{F}_\pi) \tag{A.28}$$

The plan that an agent should pursue is the most likely one.

Acknowledgements

W HAT a journey! Four years ago I embarked on this adventure, straight after the Master, with energy and enthusiasm. I still remember how Carlos suggested I apply for a PhD with him, even though he knew I did not want to pursue a doctorate. I do not recall how you actually convinced me over lunch, but I remember at the end I thought "*Why not, this PhD sounds like fun*". And it was, so thank you. Here I am now, perhaps wiser, but definitely older, still looking at the world with the same enthusiasm. The more you learn, the more you understand that there is much more to learn. This is the flame that keeps me going.

I have been lucky I had two supervisors, Carlos and Martijn, who always threw gasoline on this flame, to see what happens. You gave me the freedom to express myself and let me go my own way. Challenging me to try new things and going out of my comfort zone. Martijn, you have been an inspiration to me. Your kindness and thoughtfulness for others, your ideals, and visions against mainstream trends, profoundly impacted me and shaped who I am today. I believe the (academic and not) world should have more like you. I would also like to thank those who did not supervise me directly but were there to guide me sometimes, like Riccardo from DSCS with whom I had many interesting discussions. A special thanks to Joris as well, for the random emails to grab a coffee together. The chats we had were always fun, deep, and technical. Every time I left your office I had such great motivation to study new theories and pursue new ideas! You could challenge me with my knowledge and choices, which I really appreciate.

A PhD is a process, not a title. A process where you mainly learn from others. I will steal a quote from Mohamed, with whom I co-authored many Active Inference papers. Here it goes: "*You should surround yourself with people that are smarter than you*". So thanks to all my co-authors, colleagues, and flatmates who over the years made me ask the right questions and think. Thanks to Elia for introducing me to MPPI, and for discussing with me why robots are still so stupid and what we can do to improve! Thanks to Max for always complaining about the system, and how badly we write software in academia. I have learned many things from you, and from Chadi as well, the wizard that magically makes everything work on the robots. Without you all guys, I would not have managed to achieve what I achieved.

I have been lucky to sit at AIRLab, together with other PhDs, MSc students, and people from the industry. This made the PhD less of a lonely journey and exposed me to many different realities besides academia. It was great to work with my peers and other engineers from RoboHouse on the demos in the retail store. I have also learned a lot from the students I supervised during the years. It is not easy to be a good supervisor, and the experience with you guys made me grow considerably, thank you. So, for all the people involved in AIRLab and the ones from RoboHouse, thanks for the hard work we did together, the laughs, the ping-pong matches, and

the Friday beers. These memories are impressed in my brain and will be forever with me.

I would like to thank my parents for the support they always gave me. You would have been happy with whatever I chose to do in my life, and this is priceless because you never forced me to be someone I am not, allowing me to be where I am. I could have been a carpenter, like my grandpa, a man of different times, an artist who inspired me and keeps inspiring me, educating me to seek beauty and perfection in anything we do. Some people like you have a special place in my heart. This holds for Renzo as well, even if you are not with us anymore, I hope you know that you are the reason why I kept studying after high school in the first place. I wish you could see how far I have gotten. My sincere thanks go to you and to all the people who saw something in me.

Perhaps I should thank the odds, the series of random events and decisions that led me here. We tend to forget how lucky we are to be born in this part of the world. We have no right to complain or be unhappy.

I am grateful for all the great fun I had here at TU Delft, many friends made this experience unforgettable. A special thanks to all my bouldering buddies: struggling, failing, succeeding, and improving together was amazing. Our bouldering sessions were a nice break from my studies, together with the beers and pizzas beside the canal. Of course, this would not have been possible without you Jorge, who convinced me to start this sport. You are one of a kind, bringing mess into my organized life. Please keep the tradition of ringing the bell and showing up unannounced, spontaneous nights out in these over-optimized lives are priceless. And of course, it is thanks to you that I met Patricija. The most important person of all.

Patricija, with you and your family I had amazing experiences. You push me to be the best version of myself, to try new things, and to experience life. Every challenge with you seems like a walk in the park. From meeting and hanging out in the middle of a pandemic fighting the curfew, to randomly deciding to buy a house together in the most heated market. Nothing has ever been so easy. You supported me every day during this PhD. You made me forget all the stress and the problems that sometimes came with it, always putting a smile on my face. Perhaps more than anyone else, you see who I am, and remind me how lucky we are. Since I am with you, I cannot remember a sad day. And this says it all.

Corrado Pezzato
Delft, August 2023

Curriculum Vitæ

Corrado PEZZATO

16-04-1995, born in Mirano, Italy.

Education

2014-2017	BSc in Automation Engineering (Cum Laude)
	University of Bologna, Italy
2017-2019	MSc in Systems and Control (Cum Laude)
	TU Delft, The Netherlands
2019-2023	PhD in Cognitive Robotics
	TU Delft, The Netherlands

Awards

2023	Third prize for best paper at the ICRA 2023 workshop
	Embracing Contacts [Link]

Experience

2018-2019	President of D.S.A. Kalman at Systems and Control

List of Publications

Journals

6. Yuezhe Zhang, **Corrado Pezzato**, Elia Trevisan, Chadi Salmi, Carlos Hernández Corbato, and Javier Alonso-Mora. *Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning.* In preparation for submission at IEEE Robotics and Automation Letters (2023).

5. **Corrado Pezzato**[*], Chadi Salmi[*], Max Spahn[*], Elia Trevisan[*], Javier Alonso Mora, and Carlos Hernández Corbato. *"Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations"*. Submitted to IEEE Robotics and Automation Letters (2023).

4. **Corrado Pezzato**, Carlos Hernández Corbato, Stefan Bonhof, and Martijn Wisse. *"Active Inference and Behavior Trees for Reactive Action Planning and Execution in Robotics."* IEEE Transactions on Robotics (2023).

3. Cristian Meo, Giovanni Franzese, **Corrado Pezzato**, Max Spahn, and Pablo Lanillos. *"Adaptation through prediction: multisensory active inference torque control."* IEEE Transactions on Cognitive and Developmental Systems (2022).

2. Pablo Lanillos, Cristian Meo, **Corrado Pezzato**, Ajith Anil Meera, Mohamed Baioumy, Wataru Ohata, Alexander Tschantz, Beren Millidge, Martijn Wisse, Christopher L. Buckley, and Jun Tani. *"Active inference in robotics and artificial agents: Survey and challenges."* arXiv preprint arXiv:2112.01871 (2021).

1. **Corrado Pezzato**, Riccardo Ferrari, and Carlos Hernández Corbato. *"A novel adaptive controller for robot manipulators based on active inference."* IEEE Robotics and Automation Letters (2020).

Conferences

2. Mohamed Baioumy[*], **Corrado Pezzato**[*], Riccardo Ferrari, and Nick Hawes. *"Unbiased Active Inference for Classical Control."* In IEEE International Conference on Intelligent Robots and Systems, IROS (2022).

1. Mohamed Baioumy[*], **Corrado Pezzato**[*], Riccardo Ferrari, Carlos Hernández Corbato, and Nick Hawes. *"Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference."* In European Control Conference, ECC (2021).

---

* Indicates equal contribution

Workshops

3. **Corrado Pezzato***, Chadi Salmi*, Elia Trevisan*, Javier Alonso Mora, and Carlos Hernández Corbato. *"Sampling-Based MPC Using a GPU-parallelizable Physics Simulator as Dynamic Model: an Open Source Implementation with IsaacGym."* In Embracing Contacts Workshop at ICRA (2023).

2. Mohamed Baioumy, **Corrado Pezzato**, Carlos Hernández Corbato, Nick Hawes, and Riccardo Ferrari. *"Towards stochastic fault-tolerant control using precision learning and active inference."* In Second International Workshop on Active Inference, IWAI (2021).

1. **Corrado Pezzato**, Mohamed Baioumy, Carlos Hernández Corbato, Nick Hawes, Martijn Wisse, and Riccardo Ferrari. *"Active inference for fault tolerant control of robot manipulators with sensory faults."* In First International Workshop on Active Inference, IWAI (2020).

* Indicates equal contribution