Decision-Making in Robotics as Probabilistic Inference

A thesis submitted for the degree of

Doctor of Philosophy

Mohamed Baioumy

University College (Univ)

University of Oxford

Supervisors: Nick Hawes and Paul Duckworth

Contents

Abstract

Decision-making problems in robotics necessitate reasoning in the presence of uncertainty. Different problems in robotics often require distinct solutions. For instance, state estimation is commonly achieved by variants of the Bayes Filter, such as Kalman Filters and Particle Filters. On the other hand, planning under uncertainty is modeled using Markov Decision Processes. However, removing domain knowledge, essentially some form of probabilistic inference is being performed in most decision-making tasks. Given some random variables, some evidence, and probabilistic models, the goal is to find likely values for other random variables. In this thesis, we pose the use of probabilistic graphical models (PGMs) to model decision-making problems in robotics. We then leverage existing approaches from probabilistic inference to solve them. This provides a single coherent framework to reason about decision-making. A major contribution of this paper is assessing the utility of modelling decision-making problems as probabilistic inference. Even if it is possible to model a problem as probabilistic inference, is it *useful* to do so? Our work applies the principles of probabilistic inference to adaptive control, state estimation, fault-tolerant control, and planning under uncertainty. Furthermore, we connect our study with cutting-edge neuroscience research, particularly focusing on the free-energy principle and active inference.

I. Introduction

Intelligent agents acting in complex environments have to solve a diverse set of problems. For instance, consider a self-driving car navigating to a target destination. The tasks it solves include: localization to estimate its position, motion planning to determine which route to take, and longitudinal control to steer the vehicle. These are conceptually different tasks and their solutions differ. State-estimation is commonly achieved by variants of the Bayes Filter, such as Kalman Filters [1], Particle Filters [2], and Histogram Filters [3]. Localization and mapping algorithms build upon such approaches [4]. On the other hand, planning in uncertain environments is modelled using Markov Decision Processes (MDPs) and their variants [5]. Finally, control approaches range from classic feedback controllers [6], [7] to fault-tolerant controllers [8] and stochastic model-predictive control [9].

In all the mentioned problems, quantifying and reasoning about uncertainty is of paramount importance. The key idea in *probabilistic robotics* is to represent uncertainty explicitly. Instead of relying on a single 'best guess' as to what the ground truth might be, probabilistic algorithms represent information by probabil-

ity distributions over a whole space of possible states [3]. By doing so, they represent ambiguity and the degree of belief in a mathematically sound manner.

If the domain knowledge is removed from the mentioned problems, they become considerably similar. For instance, in localization, the aim is to estimate the state of a system given noisy sensors. Conceptually, this is very different from a control problem where the aim is to find an action given a state; since the latter involves a sense of non-determinism. However, mathematically both problems turn out to be almost equivalent. In the case of linear Gaussian systems, the Kalman filter and the Linear Quadratic controller are exactly equivalent (referred to as the Kalman Duality [10]).

Interestingly, in a large number of problems in robotics, some form of probabilistic inference is being performed: given some random variables, some evidence, and probabilistic models, the goal is to find likely values for other random variables. This raises the question, can we reason about different decision-making problems in robotics using a single coherent framework?

Additionally, a substantial body of work in computational neuroscience supports the Bayesian brain hypothesis, encompassing the free-energy principle and the active inference framework [11]–[14].

This essentially reduces to the notion that all organisms engage in some form of probabilistic inference for their decision-making. Setting aside the computational neuroscience perspective, this theory offers high-level inspiration for this thesis. Specifically: biological agents capably execute a variety of complex tasks. If their decision-making can be universally modelled using probabilistic inference, then it might also be feasible to model robotic decision-making using probabilistic inference. This leads us to the core contributions of this thesis.

A. ***Main contribution***

The main contribution of this thesis is to investigate the utility of probabilistic inference as a general framework for decision-making in robotics.

First, we examine whether probabilistic inference can be considered a general framework to model decision-making. There is a substantial body of evidence suggesting that many decision-making problems can be effectively addressed with this approach. This includes work in neuroscience [11], [15], robotics [16], [17], and control [18]. In this thesis, we do not claim that *all* decision-making problems can be resolved using probabilistic inference. Instead, we test generality by tackling a few distinct problems from different sub-fields. In Chapter III, we address classical control, adaptive control, and state-estimation. In Chapter IV, we delve into problems concerning fault detection and fault-tolerant control. In Chapter V, we explore planning under uncertainty. Each of these problems is traditionally solved using different methodologies, but in this thesis, we approach all of them using the same framework. This suggests that probabilistic inference and Probabilistic Graphical Models (PGMs) are versatile enough to tackle a variety of challenges in robotics.

Probabilistic inference can be used to address any decision-making problem [19]. However, this is somewhat of a trivial statement. One can claim that any decision-making problem can be solved using mathematics. We suggest that reasoning about decision-making problems using Probabilistic Graphical Models (PGMs) is a useful level of abstraction. This brings us to the next contribution of this thesis. We assess the utility of solving decision-making problems through PGMs. Even if it is possible to solve distinct problems using PGMs, is it useful to do so?

In this thesis, we provide evidence that PGMs and probabilistic inference can be used to tackle a variety of problems in robotics, and for every problem we investigate the utility of doing so.

Every problem addressed in this thesis is solved using the following general process.

1) A domain-specific problem is identified (e.g. fault-tolerant control for robot arm with sensory faults)
2) The domain-specific problem is formulated as a probabilistic graphical model (e.g. a factor graph).

3) Approximate or exact inference methods are used to solve for the target variable (e.g. a control action in the presence of sensory faults).

For every contribution, we highlight why following this process is useful. The unified aspect in all the papers is the process above. The domain-specific problems are different, however, all of them are modelled using PGMs and then solved using an appropriate inference method.

B. ***Thesis structure***

Chapter II provides preliminaries on the problems (e.g. fault-tolerant control) relevant to this thesis. We then highlight the potential utility of solving those problems as probabilistic inference. This is followed by three core chapters.

Chapter III focuses on control and state-estimation. In this setting, all problems consider continuous random variables. The experiments are all validated on real robots as well as simulated robots.

Chapter IV focuses on fault detection and fault-tolerant control. In this setting, most variables are continuous; however, there is a need for discrete quantities as well (is a fault present or not). The experiments are limited to simulated robots in this setting. However, where possible real-world data has been used.

Chapter V focuses on planning under uncertainty. In this setting, we focus on solving discrete Markov Decision Processes (MDPs). Although the focus is on discrete problems, we do provide notes on how this can extended to continuous and partially observable problems.

Chapter VI provides a general discussion on the utility of probabilistic graphical models and conclusions from the thesis.

C. ***List of publications***

The following is a list of publications used in this integrated thesis:

- **Mohamed Baioumy**, Corrado Pezzato, Riccardo Ferrari, Carlos H. Corbato and Nick Hawes, “Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference”, European Control Conference (ECC), 2021
- Corrado Pezzato, **Mohamed Baioumy**, Carlos H. Corbato, Nick Hawes, Riccardo Ferrari and Martijn Wisse, “Active inference for fault tolerant control of

robot manipulators with sensory faults. In International Workshop on Active Inference (pp. 20-27), Springer, 2020.

- **Mohamed Baioumy**, William Hartenmink, Riccardo Ferrari, Nick Hawes, “Beta Residuals: Improving Fault Detection and Recovery via Bayesian Inference and Precision Learning”, 11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS 2022
- **Mohamed Baioumy**, Paul Duckworth, Bruno Lacerda and Nick Hawes, “Active Inference for Integrated State-Estimation, Control, and Learning”, IEEE Intl. Conf. on Robotics and Automation (ICRA), 2021.
- **Mohamed Baioumy**, Corrado Pezzato, and Nick Hawes, “Unbiased Active Inference for Classical Control”, IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), 2022.
- **Mohamed Baioumy**, Bruno Lacerda, Paul Duckworth and Nick Hawes, “On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference”, International Workshop on Active Inference, Springer, 2021.
- **Mohamed Baioumy**, Bruno Lacerda, Paul Duckworth and Nick Hawes, “Exploiting Time-Horizon Posterior Distributions for Planning under Uncertainty”, Under review.

In addition to the publications above, below is a list of publications the student participated in that are not part of this thesis:

- Michael Painter, **Mohamed Baioumy**, Bruno Lacerda and Nick Hawes, “Monte Carlo Tree Search with Boltzmann Exploration”, NeurIPS 2023.
- **Mohamed Baioumy**, Corrado Pezzato, Carlos Hernandez Corbato, Nick Hawes and Riccardo Ferrari, “Towards Stochastic Fault-tolerant Control using Precision Learning and Active Inference”, International Workshop on Active Inference, Springer, 2021.
- Pablo Lanillos, Cristian Meo, Corrado Pezzato, Ajith Anil Meera, **Mohamed Baioumy**, Wataru Ohata, Alexander Tschantz, Beren Millidge, Martijn Wisse, Christopher L. Buckley, Jun Tani ,“Active Inference in Robotics and Artificial Agents: Survey and Challenges”, under review.
- **Mohamed Baioumy**, Matias Mattamala, Nick Hawes, “Variational inference for predictive and reactive controllers” in ICRA 2020 Workshop on New advances in Brain-inspired Perception, Interaction and Learning, Paris, France, 2020.
- **Mohamed Baioumy**, Matias Mattamala, Paul Duckworth, Bruno Lacerda and

Nick Hawes, “Adaptive Manipulator Control using Active Inference with Precision Learning,” in UKRAS20 Conference: “Robots into the real world” Proceedings, Lincoln, United Kingdom, 2020.

- Job Neven, **Mohamed Baioumy**, Wouter Wolfslag, and Martijn Wisse. “Design and evaluation of an energy-saving drive for a versatile robotic gripper.” In 2019 International Conference on Robotics and Automation (ICRA), pp. 3665-3671. IEEE, 2019.

*D. **Contributions of individual papers***

In the following, we address the contribution of each paper comprising this thesis individually.

In Chapter III, we address problems related to state-estimation and adaptive control. Section III-A presents the paper **"Active Inference for Integrated State-Estimation, Control, and Learning"**. In this paper, we explore an innovative approach for adaptive control in robotics, utilizing the Active Inference framework, a concept derived from neuroscience. This approach is adept at handling unmodeled dynamics, dampening oscillations, and maintaining robustness against poor initial parameters, as demonstrated on the 'Franka Emika Panda' 7 DoF manipulator. The approach leverages the principles of minimizing variational free-energy, commonly used in computational neuroscience. The key contributions of this work are:

- Introduction of a theoretical formulation for Active Inference that includes a temporal parameter τ, establishing a direct relationship between Active Inference and conventional control methods such as PID control.
- Development of an automatic tuning method for the controller, aligning the controller parameters with the gains typically used in traditional control setups. This innovation enhances the controller's adaptability and robustness in dynamic environments.

Section III-B presents the paper **"Unbiased Active Inference for Classical Control"**. In this work, we present an extended version of the unbiased active inference controller (u-AIC), addressing the limitations of the standard active inference controller (AIC) in robotic control. The u-AIC retains the benefits of the AIC while overcoming its drawbacks, such as biased state estimates and an implicit model of control actions. Demonstrated through simulations on a 2-DOF arm and real experiments on a 7-DOF manipulator, the u-AIC shows superior performance compared to the standard AIC. The key contributions of this paper are:

- Detailed analysis of the AIC's limitations in robot control, identifying issues like biased state-estimation and implicit control models.
- Formalization and presentation of the u-AIC, including theoretical proofs of convergence and extensions to the controller, enhancing performance and applicability in real-world robotic systems.

In Chapter IV, we address problems related to fault detection and fault-tolerant control. Section IV-A presents the paper **"Active Inference for Fault Tolerant Control of Robot Manipulators with Sensory Faults"**. In this paper, we introduce a novel fault-tolerant control scheme for robot manipulators, leveraging the active inference framework. This approach simplifies fault detection and isolation by using sensory prediction errors in free-energy calculations, eliminating the need for additional controllers for fault recovery. The scheme is demonstrated through simulations on a 2DOF robot arm, highlighting its effectiveness in managing severe sensory faults and reducing system complexity. The main contributions include:

- Development of an active inference-based fault-tolerant control scheme that streamlines the generation of residuals and thresholds for fault detection and isolation.
- Highlighting the limitations of active inference framework, and how it is fundamentally limited in the case of fault-tolerant control[1]. For instance, as will be demonstrated in the next section, consistent false positives will occur every time the goal state changes.

Section IV-B presents the paper **"Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference"**. This paper introduces a novel fault-tolerant control scheme for robotic systems, utilizing unbiased active inference. The scheme focuses on enhancing state-estimation and defining probabilistically robust thresholds for fault detection and isolation of sensory faults. This is achieved without requiring additional controllers for fault recovery. Validated on a 2-DOF manipulator, this method simplifies the fault detection process while reducing the likelihood of false positives. The main contributions of this paper are:

- Introduction of an unbiased active inference controller (u-AIC) for fault-tolerant control. This provides improved state-estimation and simplifies fault detection and isolation in robotic systems.
- Demonstration of the effectiveness of this approach through simulations, showcasing its capability to detect and recover from sensory faults with reduced false positives and enhanced robustness.

[1]Note that these limitations have been touched upon in the previous section when introducing unbiased Active inference. In the case of fault-tolerant control such limitations are more severe as presented in the discussion section of this paper.

Section IV-C presents the paper **"Beta Residuals: Improving Fault-Tolerant Control for Sensory Faults via Bayesian Inference and Precision Learning"**. This paper explores a Bayesian approach to fault-tolerant control (FTC) in robotics, specifically focusing on integrating fault detection, isolation, and accommodation into a single Bayesian inference problem. We introduce a precision-learning-based FTC technique along with a novel 'beta residual' for fault detection. This approach simplifies the FTC process by eliminating the need for explicit fault detection steps, providing implicit fault recovery, and offering interpretable information about sensor faults. The efficacy of this method is demonstrated through simulation results. The key contributions of this paper are:

- Development of a Bayesian controller for stochastic FTC that tracks naturally occurring variations and faults through precision learning.
- Introduction of the 'beta residual', providing an interpretable measure of sensor faultiness as part of the Bayesian inference process.

In chapter V, we address problems related to planning under uncertainty and reinforcement learning. Section V-A presents the paper **"On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference"**. This paper introduces an approach to solving the Stochastic Shortest-Path Markov Decision Process (SSP MDP) with an indefinite horizon through probabilistic inference. We highlight key distinctions between dynamic programming methods traditionally used in artificial intelligence for MDP solutions and active inference approaches. The main contributions of this work include:

- Development of a new algorithm for solving SSP MDPs using probabilistic inference.
- Detailed analysis of the differences between solving MDPs in the planning community and the active inference community, particularly in terms of horizon considerations, policy formation, and online versus offline planning.

Section V-B presents the paper **"Exploiting Horizon Posterior Distributions for Efficient Planning as Inference"**. In this paper, we present an Expectation-Maximization (EM) algorithm for indefinite-horizon Markov Decision Processes (MDPs) using planning as inference formulation. Our approach exploits a posterior distribution over the horizon length to optimize the policy with computational efficiency. This method contrasts with traditional dynamic programming approaches by allowing explicit

reasoning about the horizon. The main contributions of this work include:

- Introduction of a novel EM algorithm for solving indefinite-horizon MDPs, addressing problems without a fixed horizon length.
- Achievement of computational gains by leveraging the horizon posterior, which provides a principled way of determining the maximum iterations needed for optimal policy computation.

Section V-C discusses the problem of goal-based planning under uncertainty. This section highlights the utility of using PGMs for planning. Section V-D discusses sampling-based methods related to planning as inference. This is partially based on the paper **"Monte Carlo Tree Search with Boltzmann Exploration"** [2]. In this section, we discuss how sampling-based methods can used to represent beliefs (such as particle filters) and perform planning (such as Monte Carlo methods). This section is presented to contrast existing sections which rely on representing beliefs through parametric methods.

[2]Note that this paper is not included in this thesis.

II. The utility of probabilistic inference

In this thesis, we investigate the use of probabilistic inference as a general framework to tackle a variety of problems in robotics. First we briefly describe the problems relevant to this thesis. Second, we guide the reader through a framework to understand the connection between the different papers.

A. *Problems*

State-Estimation: This problem focuses on deducing the hidden state of a robot based on noisy observations. For instance, in the context of a robot arm, state-estimation involves determining the position and orientation of the arm despite uncertain sensory inputs. A common approach to tackling this issue is through the use of variants of the Kalman filter [1]–[3], [20].

Adaptive Control: Here, the challenge is to maintain effective control of a robot in the face of varying conditions or parameters. An illustrative example is a robot arm tasked with manipulating objects of different weights. The key to adaptive control is to adjust the control strategy in real-time to accommodate these varying conditions. Techniques such as model reference adaptive control are often employed to achieve this adaptability [21]–[24].

Fault-Detection and Fault-Tolerant Control: These two areas are concerned with the identification and management of faults in robotic systems. Fault detection involves recognizing when a component or system is not functioning as expected, while fault-tolerant control is about ensuring the robot continues to operate effectively despite the presence of certain faults. These aspects are crucial for the reliability and safety of robotic systems in dynamic environments [8], [25]–[27].

Planning under Uncertainty: This problem addresses the challenge of making optimal decisions in situations where the outcomes are not deterministic. It involves optimizing a reward function within a stochastic dynamic model, such as those represented by Markov Decision Processes (MDPs). Solving this problem typically involves dynamic programming methods, which provide a structured approach to making decisions over time in the presence of uncertainty [5], [28], [29].

The aim of this thesis is to test whether using Probabilistic graphical models (PGMs) and probabilistic inference (PI) is useful as a general decision-making framework in robotics. To test this assumption we have picked the above-mentioned problems.

State-estimation and control are continuous problems often approached by control researchers. Planning under uncertainty is generally discrete and tackled by different researchers. Fault-tolerant control was selected as the third area as it contains both continuous and discrete random variables.

The area of continuous control as inference, and state-estimation had the most published research. For this reason, it has less focus compared to fault-tolerant control and planning under uncertainty in this thesis. Two notable areas of research toI could have explored for the main thesis question are a) multi-agent planning and b) risk-aware control. Multi-agent problems can have interesting structures to be exploited using PGMs, and risk-aware control can naturally be modelled using probabilistic inference.

B. ***Problem setup***

As discussed, problems such as fault-tolerant control and planning are different in nature. This might make the different papers comprising the thesis seem unrelated at first glance. However, fundamentally all the methods rely on modeling and solving the problem at hand using standard methods in probabilistic inference. This section aims to guide the reader to understanding the connections between the different papers, and the utility of probabilistic inference applied to different problems in robotics.

For each paper, we discuss the mechanics involved in the problem. This includes

- **Random Variables:** Does the problem statement include continuous or discrete random variables? This distinction is crucial for determining the appropriate statistical methods and assumptions for the analysis.
- **Sub-field:** Is the problem related to state-estimation and control (S&C), Fault-tolerant control (FTC) or Planning under uncertainty?
- **Environment:** Does the section include an experiment on a robot, an experiment in simulation, or a theoretical result? This context is important to understand the practical applicability and limitations of the findings.

An overview of the papers of this thesis and the mechanics involved in solving them is summarized in Table 1.

	Subfield	Random Variable	Environment
Sec III-A	S&C	Continuous	Robot
Sec III-B	S&C	Continuous	Robot
Sec IV-A	FTC	Both	Simulation
Sec IV-B	FTC	Both	Robot
Sec IV-C	FTC	Both	Simulation
Sec V-A	Planning	Discrete	Theoretical
Sec V-B	Planning	Discrete	Simulation
Sec V-C	Planning	Discrete	Simulation
Sec V-D	Planning	Discrete	Simulation

TABLE I: Mechanics of the problems presented in each section. The column 'sub-field' refers to state-estimaiton and control (S&C), Fault-tolerant control (FTC), or planning under uncertainty.

C. *Utility of probabilistic inference*

Next to the setup of the problems and mechanics involved, we consider the potential utility of modelling the problem at hand using probabilistic inference. This includes

- **Novel Capabilities:** We develop novel capabilities compared to existing methods, this helps push the state-of-the-art in the relevant sub-field. This is not the case in all papers included in this thesis. For instance, in Section V-A we present a method to solve an SSP MDP as probabilistic inference and show its equivalence to dynamic programming approaches. This is not a novel capability as equivalent algorithm exists. However, in Section V-B we show how this formulation can allow us to reason about the planning horizon and gain computational efficiency. This is a novel capability and a contribution that pushes the state-of-the-art. Furthermore, Section III-A and [24] demonstrate that active inference can be used to develop a control strategy outperforming Model Reference Adaptive Control (MRAC). Section IV-C introduces novel methods for fault-detection and fault-tolerant control.
- **Coherence:** All aspects of the problem at hand are solved using one coherent framework. One benefit here is reducing complexity. In Section III-A, control and state-estimation are both addressed using active inference. However, learning the model parameters isn't directly approached by modelling it as a random variable, this turns out to introduce complexity in the optimization. This shows a lack of coherence. In contrast, Section IV-C achieves state-estimation, control, and learning model parameters under a single coherent framework. The same type of parameters are learned in this case. However, the optimization is greatly simplified as it can be directly included in the probabilistic inference process. Another benefit of coherence is that it unifies solutions which can lead to efficiency. For instance, consider a toolbox like GTSAM [30] which performs probabilistic inference on factor graphs. GTSAM is used to solve multiple problem such as trajectory estimation and path planning [17], [31], [32]. This means that researchers implementing solution for both problems can use the same toolbox, notation and mathematical tools. This also means that any improvement in GTSAM or new a capability translates to benefits to both strands of research. From this perspective, if there is another paradigm to implement an equivalent algorithm, it would likely be preferably to stick with probabilistic inference.
- **Joint Optimization:** Multiple problems are jointly solved instead of address-

ing each separately. Usually joint optimization brings computational efficiency. While state-estimation can be modelled as an inference problem and solved independently, and control can be similarly approached, combining both in one model for joint resolution reveals interesting properties. This is most evident in Sections IV-A, IV-B, and IV-C, where jointly performing state-estimation and fault-detection enhances performance and significantly reduces computational cost.

- **Automated Solvers:** Toolboxes for probabilistic inference (e.g. GTSAM [30] and ForneyLab [33]) can relieve the researcher from the burden of devising a custom solution to the problem at hand. This is most evident in Section IV-C where the entire stack including control, state-estimation, fault-detection and fault-recovery is addressed using an automated solver to jointly solve all problems. Additionally, Section V-C show a concrete example for how an automated solver can help the researcher focus on modelling the problem, rather the low-level optimization.
- **Connection between Literature:** Methods in diverse fields such as control theory, planning, neuroscience, and information theory are often equivalent. Recognizing these equivalences and connections allows researchers from less mature fields to leverage a wealth of knowledge from others, fostering interdisciplinary advancement.

An overview for the above potential benefit and where this is demonstrated in the thesis is in Table II.

	Novel capabilities	**Coherence**	**Joint optimization**	**Automated solvers**	**Connecting literature**
Section III-A	✓	×	✓	×	✓
Section III-B	×	✓	✓	✓	✓
Section IV-A	✓	×	✓	×	✓
Section IV-B	✓	✓	✓	×	×
Section IV-C	✓	✓	✓	✓	✓
Section V-A	×	✓	×	✓	✓
Section V-B	✓	✓	✓	✓	×
Section V-C	✓	✓	×	✓	×
Section V-D	✓	×	×	×	✓

TABLE II: Assessment of the potential utility of probabilistic inference as applied to every paper in this thesis.

In every section where a paper is present, we will revisit the above. We will highlight the mechanics of the inference problem at hand, and what utility is arguably gained by using probabilistic inference.

Note that the above points are related, but distinct. For example, joint optimization using an automated solver would be possible if coherence applies to the whole stack. This is the case in IV-C, thus satisfying all three properties. However, joint optimization is possible regardless of whether coherence is satisfied or not. In Section IV-A, Fault-detection and isolation (FDI) is not modelled as an inference problem, yet joint optimization is performed and computation gains are realized. In Section IV-C on the otherhand, FDI is modelled as inference and again joint optimization yields computational gains. This highlights that joint optimization can be beneficial regardless of coherence. Similarly, one can utilize automated solvers to solve each problem independently. In this case, automated solvers apply, but there is no joint optimization. The most basic illustration of this is when there is a single problem. This can lead to efficiency (e.g. V-C), however, no joint optimization is performed.

III. Control and state-estimation

This section discusses work on utilizing probabilistic inference for control and state-estimation.

A. *Integrated state-estimation, control, and learning using active inference*

This work presents an approach for adaptive control in robotic manipulators, employing the Active Inference framework from computational neuroscience. Demonstrated on the 'Franka Emika Panda' 7 DoF manipulator, this method shows promise in handling unmodeled dynamics, damping oscillations, and maintaining robustness against initial parameter inaccuracies. The paper's key contributions include a novel theoretical formulation for Active Inference with a temporal parameter τ, establishing a clear connection to traditional control methods such as PID control, and the development of an automated tuning technique for controller parameters, thereby enhancing adaptability and robustness in dynamic environments.

This paper is concerned with state-estimation and control (S&C), it deals with continuous random variables and it's validated on a real robot as well as simulated robots.

Now we discuss the potential utility of utilizing probabilistic inference in this paper.

- **Novel Capabilities (✓):** This paper (along with [24]) demonstrates that active inference can be used to develop a control strategy outperforming Model Reference Adaptive Control (MRAC). Our work can deal with unmodelled dynamics and is equivalent to a self-turning PID controller. This is a novel capability as we outperform a state-of-the-art approach.
- **Coherence (×):** This paper does not show coherence. In this paper, state-estimation and control are achieved by minimizing free-energy. This means we can use a standard gradient descent method to solve both problems. However, learning model parameters was not explicitly modelled as probabilistic inference. This means that as a researcher we had to manually account for considerations for how to learn model parameters. For instance in the fifth section of the paper, we show how we accounted for complexity to estimate a co-variance value. Co-variance is a strictly positive property (and a co-variance matrix is positive semi-definite) and thus we need to account for that. In contrast to this paper, Section IV-C presents an approach where one is also learning model

parameters. However in that case, everything is modelled as a random variable and inference is performed to achieve all functionality. In Section IV-C we thus satisfy coherence, as a result all complexity for optimizing the co-variance is taken care of by a standard solver. However, in this paper, we had to manually account for such complexity.

- **Joint Optimization (✓) :** As indicated by the title of this paper, all problems are jointly optimized. We show that the problems are intertwined as well. One cannot separate the state-estimation from the control module in the current algorithm. This is key to its self-tuning ability. Jointly optimization is thus a necessity to achieve the state-of-the-art results.
- **Automated Solvers (×):** This paper did not fully rely on automated solvers. We manually implemented the method to optimize the hyperparameters of this algorithm. Since this was not fully modelled as a probabilistic inference problems, a standard solver couldn't directly be used. However, minimizing free-energy for control and state-estimation was achieved using a standard solver.
- **Connection to Literature (✓) :** This paper connects active inference (a method from computational neuroscience) to PID control (a classical control method).

The paper starts on the following page.

Active Inference for Integrated State-Estimation, Control, and Learning.

Mohamed Baioumy Paul Duckworth Bruno Lacerda Nick Hawes

***Abstract*— This work presents an approach for control, state-estimation, and learning model (hyper)parameters for robotic manipulators. It is based on the active inference framework, prominent in computational neuroscience as a theory of the brain, where behaviour arises from minimizing variational free-energy. First, we show there is a direct relationship between active inference controllers and classic methods such as PID control. Second, we demonstrate its application for adaptive and robust behaviour of a robotic manipulator that rivals state-of-the-art. Additionally, we show that by learning specific hyperparameters, our approach can deal with unmodeled dynamics, damps oscillations, and is robust against poor initial parameters. The approach is validated on the 'Franka Emika Panda' 7 DoF manipulator. Finally, we highlight the limitations of active inference controllers for robotic systems.**

Supplementary Material

`https://github.com/MoBaioumy/active_inference_panda_paper`.

I. INTRODUCTION

It is crucial for intelligent robots to be able to adapt to the presence of unmodeled dynamics. This is necessary for applications such as aerial vehicles encountering unpredicted wind dynamics [1], manipulators handling objects of unknown weight [2], and autonomous vehicles on slippery road surfaces [3]. Humans are skilled in this regard and recent work in robotics has taken inspiration from *active inference* [4], a neuroscientific theory of the brain and behaviour. Active inference provides a framework for understanding decision-making of biological agents where optimal behavior arises from minimising variational free-energy: a measure of the fit between an internal model and (past) sensory observations. Additionally, agents act in a fashion that fulfills prior beliefs about preferred future observations [5]. This framework has been employed to explain and simulate a wide range of complex behaviors including abstract rule learning, speech generation, planning, and navigation [6], [7], [4].

The active inference controller (AIC) [8], allows an agent to jointly perform state-estimation and control by minimizing a single quantity: variational free-energy. Subsequent work has utilized the AIC in robotics [9], [10], [11]. The control and state-estimation parts are *coupled* and the AIC architecture is unusual compared to methods commonly used in robotics. Usually, when performing state-estimation (for instance using a Kalman or particle filter [12], [13]) observations are required. The aim is then to find a belief state that is close to the true (hidden) state. Additionally, controllers (such as a PID [14]) require a target state along the current state. A PID controller then provides a control law to reach the target from the current state. In the AIC however, state-estimation requires the target and the observations which is unusual from a control perspective; those are used to produce a 'biased' belief. The control part of the AIC requires the observation and the 'biased' belief. It is thus unclear how to systematically tune such a controller (since tuning the controller affects state-estimation). Furthermore, even slight changes to the parameters of the state-estimation component could cause the controller to be unstable.

Authors are with the Oxford Robotics Institute, University of Oxford, United Kingdom. For correspondence: {mohamed, pduckworth, bruno, nickh}@robots.ox.ac.uk

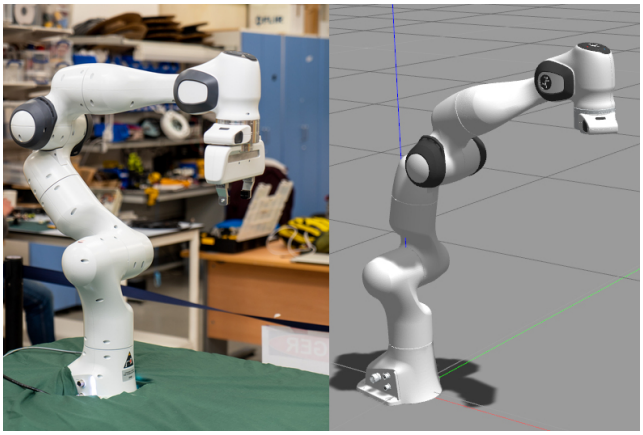


Fig. 1: The Franka Emika Panda 7 DoF manipulator real robot (left) and simulated robot in Gazebo (right).

This paper has three contributions. First, we present a formulation that includes a temporal parameter τ. Second, we highlight theoretical results showing the exact relationship between our formulation and existing methods. For example, if our introduced parameter τ approaches zero, the approach converts to a classic PID controller. When $\tau \rightarrow \infty$, the approach converts to a filter. If τ is set to the unity, the formulation presented in [8] is recovered. The third contribution is presenting a method to automatically tune the controller. We show that the gains of the controller correspond to the covariances of the observation model and that learning the optimal covariances results in learning the optimal gains for the controller. However, the covariances do not converge when the system reaches the target. We thus present an alternative by learning the introduced temporal parameter τ. This reduces oscillations and improves robustness.

As a running example throughout the paper, the mass-spring-damper [15] system is used. However, we validate the approach on a 'Franka Emika Panda' 7 DoF manipulator in the results section.

II. Related work

There are only a few attempts to use active inference in robotics and control. In [16] a simulated PR2 robot is controlled by open-loop active inference but the computational complexity made an online implementation unfeasible. In [17], the authors use free-energy minimization for adaptive body perception. The same authors extended their work to include control in [9]. In [10] an implementation of active inference is presented with real hardware on a humanoid robot capable of performing reaching behaviors with both arms along with active head object tracking in the presence of noisy observations. This work is extended in [18] using deep neural networks as function approximators.

In [11] an approach for control of robotic manipulators is presented. Adaptive behaviour is demonstrated against the state-of-the-art model-reference adaptive control (MRAC) [1]. Additionally, in [21], the authors use active inference to achieve fault-tolerant control for sensory faults [22], [23]. The robot can detect whether its sensors are faulty and the control law is automatically adjusted to account for the detected faults. In [24], free-energy minimization is used to improve state-estimation under coloured noise. In [25], [26], [27] active inference with factor graphs is proposed as an effective control method; however, results are provided only for a simulated toy example.

III. Active Inference framework

This section introduces active inference as a general framework and derives the key equations for free-energy (F). Free-energy is used in later sections to achieve state-estimation, control, and hyperparameter learning.

A. Variational free energy

We consider an agent in a dynamic environment that receives observation o about the hidden state s. Given a model of the agent's world, Bayes' rule can be used to find $p(s|o)$. However, the normalization term $p(o)$ in Bayes' rule involves calculating an integral making calculations of all but trivial examples infeasible. Instead, the agent can approximate the posterior distribution $p(s|o)$ with a *variational distribution* q over states, which we can define to have a simpler form (such as a Gaussian). The goal is then to minimize the difference between the two distributions. The mismatch between the two distributions can be computed using the Kullback Leibler (KL) divergence [28], [29]:

$$\begin{aligned} KL(q(s)||p(s|o)) &= \int q(s) \ln \frac{q(s)}{p(s,o)} ds + \ln p(o) \\ &= F + \ln p(o). \end{aligned} \tag{1}$$

The quantity F is referred to as the (variational) free-energy and minimizing F minimizes the KL-divergence. F is also often referred to as the evidence lower bound (ELBO) in the Machine Learning community[2]. If we choose $q(s)$ to be a Gaussian distribution with mean μ, and utilize the Laplace approximation [30], the free-energy expression from Equation 1 simplifies to:

$$F \approx -\ln p(\mu, o). \tag{2}$$

The expression for F is solely dependent on one parameter, μ, which is referred to as the 'belief state' or simply 'belief'. The objective is to find the μ which minimizes F and thus minimizes the divergence between $q(s)$ and $p(s|o)$. For a robotic manipulator, the set of observations ($\mathbf{o}$) and beliefs ($\boldsymbol{\mu}$) are vectors with length depending on the number of degrees of freedom.

Generalised motions (GM) [31] are used to represent the (belief) states of a dynamical system, using increasingly higher-order derivatives of the system state. This means that the n-dimensional state $\boldsymbol{\mu}$ and its higher order time derivatives are combined in $\tilde{\boldsymbol{\mu}}$ (i.e. $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}, \boldsymbol{\mu'}...]$). Similarly, observations are combined as $\tilde{\mathbf{o}} = [\mathbf{o}, \mathbf{o}' \ ...]$. In the context of a robotic manipulator, $\mathbf{o}$ may represent the sensory observation of joint positions, while $\mathbf{o}'$ represents the observation of joint velocities. Similarly, $\boldsymbol{\mu}$ represents the belief about all joint positions and $\boldsymbol{\mu'}$ the joint velocities.

B. Observation model and state transition model

Taking generalized motions into account, the joint probability from Equation (2) can be written as:

$$p(\tilde{\mathbf{o}}, \tilde{\boldsymbol{\mu}}) = p(\tilde{\mathbf{o}}|\tilde{\boldsymbol{\mu}}) p(\tilde{\boldsymbol{\mu}}) = p(\mathbf{o}|\boldsymbol{\mu}) p(\mathbf{o}'|\boldsymbol{\mu'}) p(\boldsymbol{\mu'}|\boldsymbol{\mu}) p(\boldsymbol{\mu''}|\boldsymbol{\mu'}), \tag{3}$$

where $p(\mathbf{o}|\boldsymbol{\mu})$ is the probability of receiving an observation $\mathbf{o}$ while in (belief) state $\boldsymbol{\mu}$, and $p(\boldsymbol{\mu'}|\boldsymbol{\mu})$ is the state transition model (also referred to as the dynamic model or the generative model). The state transition model predicts the state evolution given the current state. These distributions are assumed Gaussian according to:

$$\begin{aligned} p(\mathbf{o}|\boldsymbol{\mu}) &= \mathcal{N}(g(\boldsymbol{\mu}), \Sigma_o), \ \ p(\mathbf{o}'|\boldsymbol{\mu'}) = \mathcal{N}(g'(\boldsymbol{\mu'}), \Sigma_{o'}), \\ p(\boldsymbol{\mu'}|\boldsymbol{\mu}) &= \mathcal{N}(f(\boldsymbol{\mu}), \Sigma_\mu), \ \ p(\boldsymbol{\mu''}|\boldsymbol{\mu'}) = \mathcal{N}(f'(\boldsymbol{\mu'}), \Sigma_{\mu'}), \end{aligned} \tag{4}$$

where the functions $g(\boldsymbol{\mu})$ and $g'(\boldsymbol{\mu'})$ represent a mapping between observations and states. For many applications in robotics, the state is directly observable. For instance, in the context of a robotic manipulator the state consists of the positions and velocities of all joints, and the manipulator is provided with position and velocity encoders. Thus we assume: $g(\boldsymbol{\mu}) = \boldsymbol{\mu}$ and $g'(\boldsymbol{\mu'}) = \boldsymbol{\mu'}$. The functions $f(\boldsymbol{\mu})$ and $f(\boldsymbol{\mu'})$ represent the evolution of the belief state over time. We encode the agent's target state $\boldsymbol{\mu_d}$ in $f(\boldsymbol{\mu})$. Our contribution is to introduce τ as part of these funcitons: $f(\boldsymbol{\mu}) = (\boldsymbol{\mu_d} - \boldsymbol{\mu})\tau^{-1}$ and $f'(\boldsymbol{\mu'}) = \tau^{-1}\boldsymbol{\mu'}$, where $\boldsymbol{\mu_d}$ is the desired state and τ a time scale (explained in Section IV-D).

[1] Model-reference adaptive control [19], finds a control law that will guide the system to behave as specified by a chosen reference model. Another common method used for adaptive control is 'self-tuning adaptive control', which represents the robot as a linear discrete-time model and estimates the unknown parameters online, substituting them in the control law [20].

[2] Minimizing the ELBO and thus the KL divergence is common in variational inference, a method for approximating probability densities [28].

To simplify notation, we define the following error terms: $\boldsymbol{\varepsilon_\mu} = \boldsymbol{\mu'} - (\boldsymbol{\mu_d} - \boldsymbol{\mu})\tau^{-1}$, $\boldsymbol{\varepsilon_{\mu'}} = \boldsymbol{\mu''} + \tau^{-1}\boldsymbol{\mu'}$, $\boldsymbol{\varepsilon_o} = \mathbf{o} - \boldsymbol{\mu}$, $\boldsymbol{\varepsilon_{o'}} = \mathbf{o'} - \boldsymbol{\mu'}$. Now that all the terms have been defined, F can be expanded to:

$$F = \frac{1}{2}\sum_i \left(\boldsymbol{\varepsilon}_i^\top \Sigma_i^{-1}\boldsymbol{\varepsilon}_i + \ln|\Sigma_i|\right) + C, \tag{5}$$

where $i \in \{o,\ o',\ \mu,\ \mu'\}$, C is a constant, and the covariance matrices are defined in Equation 4. Equation (5) differs from the work presented in [11], [10] in the terms with $\ln|\Sigma|$, which are not explicitly included but rather added to the constant. The significance of this difference is highlighted in Section V-A.

IV. State-estimation and Control

We now introduce how to perform state-estimation and control by minimizing F. We show how the estimation step biases the belief towards the goal state. The control step then steers the system from its observation $\mathbf{o}$ to its (biased) belief $\boldsymbol{\mu}$. In Section IV-E we show that if $\tau^{-1} \to \infty$, the approach converts to a classic PID controller. Additionally, if $\tau^{-1} \to 0$, the approach reduces to a filter.

A. State estimation by minimizing free-energy

Estimating the state of our system is achieved by finding a value $\boldsymbol{\mu}$ that minimizes F. Gradient descent is a simple way to accomplish that:

$$\dot{\tilde{\boldsymbol{\mu}}} = D\tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial F}{\partial \tilde{\boldsymbol{\mu}}}, \tag{6}$$

where κ_μ is a tuning parameter and D is temporal derivative operator ($D\boldsymbol{\mu} = \boldsymbol{\mu'}$). The dot refers to the difference between two time-steps ($\dot{\boldsymbol{\mu}} = \boldsymbol{\mu}[t+1] - \boldsymbol{\mu}[t]$). Using Equation (6) the agent takes one step in the gradient descent at every time-step. In this case, the equation expands to:

$$\begin{aligned}
\dot{\boldsymbol{\mu}} &= \boldsymbol{\mu'} + \kappa_\mu \Sigma_o^{-1}\boldsymbol{\varepsilon_o} - \tau^{-1}\kappa_\mu \Sigma_\mu^{-1}\boldsymbol{\varepsilon_\mu} \\
\dot{\boldsymbol{\mu'}} &= \boldsymbol{\mu''} + \kappa_\mu \Sigma_{o'}^{-1}\boldsymbol{\varepsilon_{o'}} - \kappa_\mu \Sigma_\mu^{-1}\boldsymbol{\varepsilon_\mu} - \tau^{-1}\kappa_{\mu'}\Sigma_{\mu'}^{-1}\boldsymbol{\varepsilon_{\mu'}} \\
\dot{\boldsymbol{\mu''}} &= -\kappa_{\mu'}\Sigma_\mu^{-1}\boldsymbol{\varepsilon_{o'}}
\end{aligned} \tag{7}$$

The first equation states that belief is refined using the term $\kappa_\mu \Sigma_o^{-1}\boldsymbol{\varepsilon_o}$ which moves our new belief towards the value just observed. Additionally, the term $\tau^{-1}\kappa_\mu\Sigma_\mu^{-1}\boldsymbol{\varepsilon_\mu}$, shifts the belief towards the target $\boldsymbol{\mu_d}$ since $\boldsymbol{\varepsilon_\mu} = \boldsymbol{\mu'} - (\boldsymbol{\mu_d} - \boldsymbol{\mu})\tau^{-1}$. Essentially, this 'biases' the belief towards preferred future states (target state $\boldsymbol{\mu_d}$). The degree to which the system is biased depends on the values of τ^{-1} and Σ_μ^{-1}.

B. Control by minimizing free-energy

Next to state-estimation, the agent can apply an action $\ominus$ to the environment. In the context of a robotics manipulator, this is a vector containing torques of all joints. To find the control action which minimizes F, gradient descent is used:

$$\dot{\mathbf{a}} = -\kappa_a \frac{\partial F}{\partial \mathbf{a}} = -\kappa_a \frac{\partial F}{\partial \tilde{\mathbf{o}}}\frac{\partial \tilde{\mathbf{o}}}{\partial \mathbf{a}}, \tag{8}$$

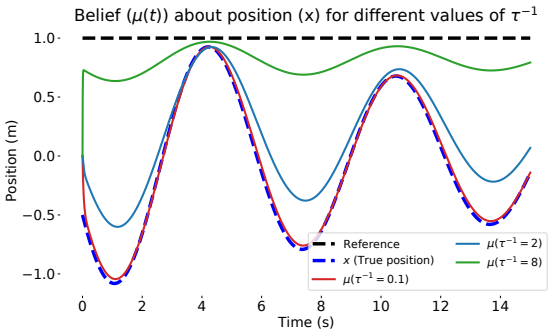


Fig. 2: Separately performing state-estimation for different values of τ^{-1}. Higher values of τ^{-1} result in more bias.

where κ_a is a tuning parameter. The term $\frac{\partial \tilde{\mathbf{o}}}{\partial \mathbf{a}}$ is assumed linear, and equal to the identity matrix (multiplied by a constant) similar to existing work [11], [10]. Actions are then computed as:

$$\dot{\mathbf{a}} = -\kappa_a(\Sigma_o^{-1}(\mathbf{o} - \boldsymbol{\mu}) + \Sigma_{o'}^{-1}(\mathbf{o'} - \boldsymbol{\mu'})). \tag{9}$$

This controller essentially steers the system from its observed state $\mathbf{o}$ to its (biased) belief $\boldsymbol{\mu}$.

Note how the current control law does not contain any information about the dynamical system, it is thus a reactive controller. The control law only requires $\mathbf{o}$ and $\boldsymbol{\mu}$. This controller thus operates in the presence of unmodeled dynamics similar to a PID controller.

C. Simultaneous state-estimation and control

Our approach performs state estimation and control simultaneously. The estimation and control steps are dependent. This is because the estimation step biases the belief $\boldsymbol{\mu}$ towards the target $\boldsymbol{\mu_d}$. The controller then steers the system from the observation $\mathbf{o}$ to the biased estimated state $\boldsymbol{\mu}$. If τ^{-1} and Σ_μ^{-1} are large, the belief $\boldsymbol{\mu}$ is biased more towards the target $\boldsymbol{\mu_d}$.

To illustrate this, consider the mass-spring-damper system given by the equation: $\ddot{x} = a(t) - k_1 x - k_2 \dot{x}$, where s is the position of the mass, $a(t)$ the control action, k_1 the spring constant (set to $1N/m$), k_2 the damping coefficient (set to $0.1Ns/m$) and the system has unit mass. It's simulated with initial conditions $x(0) = -0.5m$, $\dot{s}(0) = -1m/s$ and $a(t) = 0N$. Equations (7) are used to perform state estimation. To challenge the system, the initial beliefs are inaccurate ($\boldsymbol{\mu}(0) = 0m$ and $\boldsymbol{\mu'}(0) = -1.5m/s$). The system is simulated for different values of τ^{-1} and presented in Figure 2.

It is clear that higher values of τ^{-1} provide more bias towards the target. For $\tau^{-1} = 8$ (green line in Figure 2), the estimate is close to the target (black dashed line) and far away from the actual position (blue dashed line) as opposed to setting $\tau^{-1} = 0.1$ (red line), the belief is closer to the real trajectory (the latent state). If $\tau^{-1} \to 0$, the estimation step reduces to a pure estimator, which would follow the trajectory without any bias towards the target.

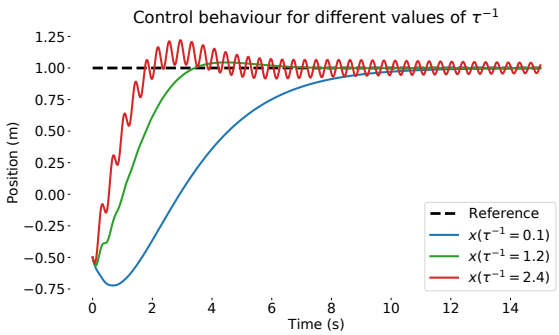


Fig. 3: True position (x) for different values of τ^{-1}. Higher values of τ^{-1} provide more bias towards the target and thus more aggressive control causing overshoot oscillations.

Enabling control steers the system to its target. The τ^{-1} in this case affects how aggressive the controller is. Larger values give more bias towards the target, the term $(\mathbf{o} - \boldsymbol{\mu})$ is larger, and thus the controller is more aggressive. Figure 3 shows an illustration for different values of τ^{-1}.

D. Understanding the temporal parameter τ

The generative model specified by Equations (3) and (4) include the function $f(\boldsymbol{\mu})$ which determines how the belief state evolves over time i.e. $f(\boldsymbol{\mu}) = (\boldsymbol{\mu_d} - \boldsymbol{\mu})\tau^{-1}$.

The belief state is specified to evolve over time as the derivative between the current belief $\boldsymbol{\mu}$ and target $\boldsymbol{\mu_d}$. This can be evaluated as the ($\boldsymbol{\mu_d}$ - $\boldsymbol{\mu}$) divided by a time scale τ. The smaller τ, the larger the derivative. If τ approaches zero ($\tau^{-1} \to \infty$), the value $f(\boldsymbol{\mu})$ approaches ∞. As a result, the belief is infinitely biased towards the target and $\boldsymbol{\mu} \approx \boldsymbol{\mu_d}$.

E. Relationship to a classic PID Controller and filters

A classic PID controller defines an error term $e = (\boldsymbol{\mu_d} - \mathbf{o})$. The control action is then chosen as:

$$a = P \cdot e + I \int e dt + D \frac{de}{dt},$$

where P, I and D are tuning parameters.

For the control law defined by active inference, our $(\mathbf{o} - \boldsymbol{\mu})$ is similar to the error term. Additionally, as explained in the previous section, when $\tau^{-1} \to \infty$ then $\boldsymbol{\mu} \approx \boldsymbol{\mu_d}$. Now the control law of active inference can be rewritten in terms of the error term as:

$$\dot{a} = \kappa_a \Sigma_o^{-1} e + \kappa_a \Sigma_{o'}^{-1} \frac{de}{dt}.$$

This means that if $\tau^{-1} \to \infty$, the active inference controller is equivalent to a PI Controller i.e. PID with $D = 0$, a P gain of $\kappa_a \Sigma_{o'}^{-1}$ and an I gain of $\kappa_a \Sigma_o^{-1}$. If one considers the generalized motions (section III) up to a third order, the control law would include a non-zero D term.

The relationship to a pure estimator (plain filter) is straightforward. As previously mentioned, if $\tau^{-1} \to 0$, the estimation step reduces to a filter. Essentially, this indicates that the estimation step has zero bias towards the target. As Figure 2 shows, for very small values of τ^{-1}, the belief is close to the latent state (true position) without bias.

V. Learning the Hyperparameters

We have shown that state estimation and control can be performed using gradient descent on the free-energy F. In addition to using gradient descent of the free energy for state-estimation and control, we apply it to learn the hyperparameters online. Previous work such as [11] does not update the hyperparameters online which makes the performance extremely sensitive to initialization, prone to instabilities, and reduces overall performance.

A. Learning model variances

As illustrated in Section IV-E, the model variances Σ_o^{-1} and $\Sigma_{o'}^{-1}$ can be considered as gains for the controller, similar to the 'P' and 'I' gains in a PID controller. Additionally, the values Σ_μ^{-1} and $\Sigma_{\mu'}^{-1}$ affect how much the estimation step biases the controller towards the desired position.

We can update Σ_o and $\Sigma_{o'}$ using gradient decent on F as:

$$\dot{\Sigma}_o = -\kappa_\sigma \frac{\partial F}{\partial \Sigma_o}, \quad \dot{\Sigma}_{o'} = -\kappa_\sigma \frac{\partial F}{\partial \Sigma_{o'}}. \tag{10}$$

The presented update rules have several practical issues. First, in any high dimensional case, Σ_o would be a matrix. Since in most equations presented so far, the inverse Σ_o^{-1} is used, updating the covariance matrix using Equations 10 then inverting it would be computationally expensive. A workaround is to simply update the inverse covariance matrix (the precision matrix), as done in [32]:

$$\dot{\Sigma}_o^{-1} = -\kappa_\sigma \frac{\partial F}{\partial \Sigma_o^{-1}}, \quad \dot{\Sigma}_{o'}^{-1} = -\kappa_\sigma \frac{\partial F}{\partial \Sigma_{o'}^{-1}}. \tag{11}$$

Additionally, a lower bound on the diagonal elements is set to keep the matrix positive semi-definite as suggested in [33].

We demonstrate the effect of updating the covarince by keeping $\Sigma_{\mu'}^{-1}$ fixed at 0.5 and Σ_μ^{-1} will be varied. If Σ_μ^{-1} is too high, the system suffers from oscillations and overshoot. However, if Σ_o^{-1} and $\Sigma_{o'}^{-1}$ are updated during run-time, the controller shows improved behaviour. Results are shown in Figure 4. The convergence of Σ_o^{-1} occurs when $\frac{\partial F}{\partial \Sigma_o^{-1}} = 0$. Since the observations change and have a certain level of noise, Σ_o converges to the expected value of $\boldsymbol{\varepsilon_o \varepsilon_o^\top}$. This does not necessarily happen upon reaching the target state.

B. Learning the temporal parameter

We previously showed the importance of choosing appropriate values for τ^{-1}: If the value is too high, the controller suffers from overshoot and oscillations, see Figure 3. On the other hand, a low value results in a slow response. Ideally, the value for τ would be high towards the start but decrease as the system reaches the target. This is the value that minimizes F and can be found using gradient descent on F as:

$$\frac{\partial F}{\partial \tau^{-1}} = -2\Sigma_{\mu'} \varepsilon_\mu (\boldsymbol{\mu_d} - \boldsymbol{\mu}) + 2\Sigma_{\mu'} \varepsilon_{\mu'} \boldsymbol{\mu'}. \tag{12}$$

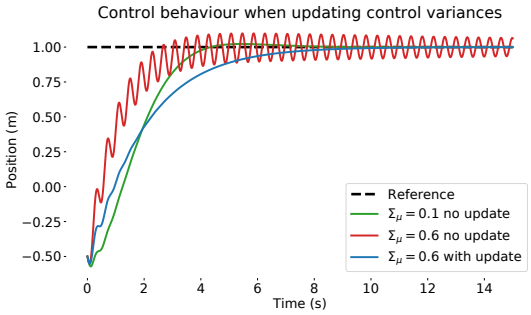


Fig. 4: Effect of updating the the control variances (Σ_o^{-1} and $\Sigma_{o'}^{-1}$). When the value of Σ_μ^{-1} is initialized at high values, the system oscillates. Updating Σ_o^{-1} and $\Sigma_{o'}^{-1}$ essentially 'tunes' the controller and ensures a robust performance.

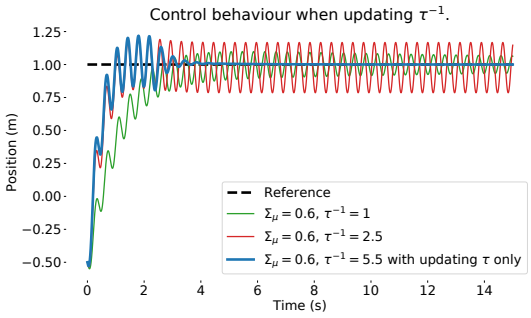


Fig. 5: Effect of updating the value of τ^{-1} during operation. This figure shows that increasing both Σ_μ or τ^{-1} results in overshoots and severe oscillations. Rather than tuning Σ_o^{-1} and $\Sigma_{o'}^{-1}$, updating τ^{-1} can be sufficient.

Note how the inverse τ^{-1} is updated rather than τ directly. Similar to the variances, all previous equations contain τ^{-1} and since inverting a matrix is computationally expensive, the inverse is directly updated. Additionally, τ^{-1} requires us to define a lower bound. The optimization can result in τ^{-1} approaching zero which means the controller converts to a pure estimator. In this work, τ^{-1} is set to have a minimum value of 0.5 on the diagonal elements and zero elsewhere.

Using Equation 12, the oscillations can be damped as well as improving settling time as shown in Figure 5. Note how updating τ^{-1} only is satisfactory to eliminate the oscillations (no update of Σ_o^{-1} or $\Sigma_{o'}^{-1}$ is necessary). The convergence of τ^{-1} occurs at $\boldsymbol{\mu_d} = \boldsymbol{\mu}$ and $\boldsymbol{\mu'} = 0$ which corresponds to the controller settling at its target position. The updates for τ will thus retune the controller appropriately until the target is reached. This provides a preference for updating τ rather than updating Σ_o^{-1} or $\Sigma_{o'}^{-1}$ in most cases.

C. Limitations

In the active inference setting, the belief is intentionally biased towards the target to achieve control. However, the belief is biased and thus not accurate unless the system reaches the target. If an accurate belief is required for other parts of the robotic system, this could be problematic.

Additionally, the terms Σ_o^{-1} and $\Sigma_{o'}^{-1}$ are defined as part of the sensory model. In the context of a filter, this would indicate the level of Gaussian noise affecting the position and velocity encoders. However, since the beliefs are biased, Σ_o^{-1} and $\Sigma_{o'}^{-1}$ converge to a value that is much higher than the actual Gaussian noise affecting the system. The values for these variables therefore lose their physical meaning. This bias can also cause false positives when reasoning about the accuracy of the sensor. Work in [21] explores this problem.

Finally, since estimation and control are dependent on each other, it is not straight-forward to tune all the parameters separately. Learning the parameters Σ_o^{-1} and $\Sigma_{o'}^{-1}$ boosts the performance as shown in the previous section. However, if Σ_μ^{-1} and $\Sigma_{\mu'}^{-1}$ are also updated, the performance decreases significantly.

VI. Results on a Robotic Manipulator

This section evaluates the presented approach against the active inference controller (AIC) from [11]. We show that our approach outperforms the AIC from [11] at the task of manipulating different payloads and setting different initial parameters for the variances and different values of τ. More results (including time-varying target states) can be found in our other work [32], [34].

A. Robustness of initial settings

The AIC from [11] achieves adaptive control without an explicit dynamics model. However, it is sensitive to the initialization of its parameters. By slightly changing Σ_μ^{-1} for instance, the system suffers from severe oscillations and never converges to the target state.

If the AIC is tuned optimally ($\Sigma_o^{-1} = 1.5I$, $\Sigma_{o'}^{-1} = 0.5I$, $\Sigma_\mu^{-1} = 0.1I$ and $\Sigma_{\mu'}^{-1} = 0.5I$), this results in satisfactory behaviour. In this case, I refers to the 7x7 identity matrix. However, if we vary $\Sigma_\mu^{-1} = 0.1I$ or to other values ($0.3I$ and $0.5I$), the performance gets considerably worse. In our approach, we update τ, Σ_o and $\Sigma_{o'}$ online to retune the controller. We ran the experiment for a pick-and-place task (as in [11]) for several values of Σ_μ^{-1} and recorded in Table I the Mean Absolute Error (MAE) defined as:

$$MAE = \frac{1}{n_t}\sum_{j=1}^{n_t} |\boldsymbol{\mu_d} - \boldsymbol{\mu}|.$$

When the AIC is properly tuned $\Sigma_\mu^{-1} = 0.1I$, the two cases have the same MAE (tuning does not matter since the initialization was optimal). However, when $\Sigma_\mu^{-1} = 0.3I$, the controller becomes does not converge to the target state (visualized in Figure 6) . The MAE increases to more than triple its value while in the case of tuning the hyperparameters, the MAE actually decreases. This is due to the fact that increasing Σ_μ^{-1} makes the controller more aggressive and when tuned, it does not oscillate and also has a slightly faster response. In a similar fashion, results for changing the value of τ^{-1} are recorded in Table II. Again, the MAE is much lower when using our method.

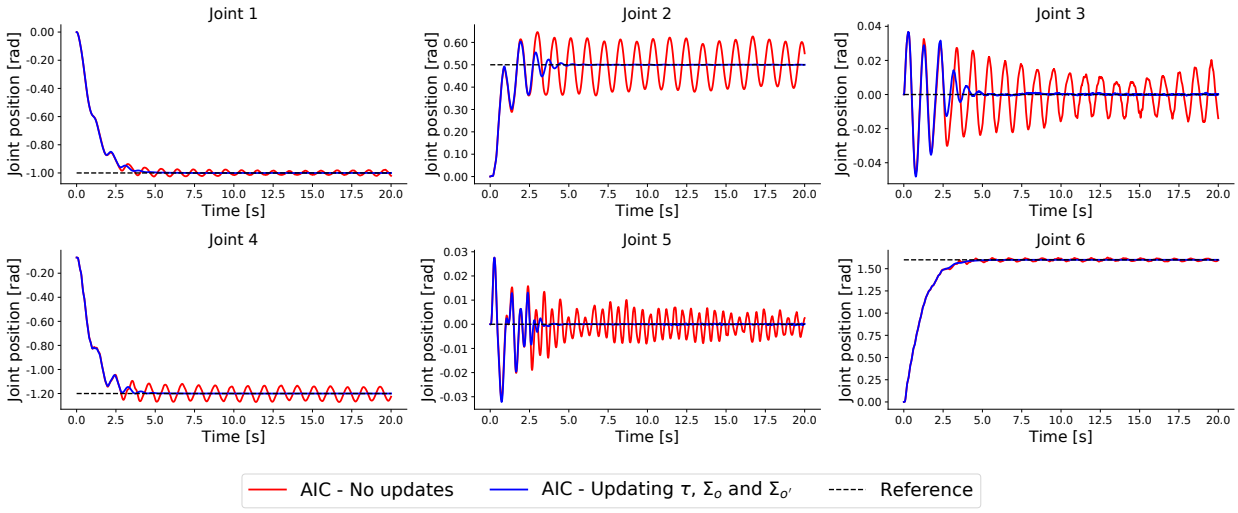


Fig. 6: Step response for the 6 joints of the robot arm with and without updating hyperparameters in the AIC. This graphs corresponds to the second column of Table I ($\Sigma_\mu^{-1} = 0.3I$). It is clear that updating the hyperparameters reduces oscillations.

B. Results on adaptive behaviour

For the last experiment, the robot carries varying payloads. All controllers are tuned to have identical performance in the no-payload case. Subsequently, we test three different payloads: $m = 1kg$, $m = 2kg$, and $m = 3kg$ (max payload for the Panda arm). The MAE for these cases is recorded in Table III.

	$\Sigma_\mu^{-1} = 0.1I$	$0.3I$	$0.5I$
No updates	0.028	0.088	0.118
Updating τ^{-1}, Σ_o^{-1} and $\Sigma_{o'}^{-1}$	0.028	**0.025**	**0.032**

TABLE I: MAE for different values of Σ_μ^{-1}.

	$\tau^{-1} = 2I$	$\tau^{-1} = 3I$
No updates	0.091	0.123
Updating τ^{-1}, Σ_o^{-1} and $\Sigma_{o'}^{-1}$	**0.025**	**0.032**

TABLE II: MAE for different values of τ^{-1}.

	$m = 1kg$	$m = 2kg$	$m = 3kg$
AIC no updates	0.024	0.029	0.027
AIC with updates (ours)	**0.020**	**0.020**	**0.021**

TABLE III: Mean Absolute Error (MAE) for different payloads in case of updating hyperparameters and no updates.

VII. Discussion and Future Work

In the AIC, actions are not explicitly modeled. In Section III, the generative model was selected to have the form $p(\mathbf{o}, \boldsymbol{s})$ which does not explicitly include any notion of an action $\mathbf{a}$. Thus to choose the action that minimizes free-energy, the chain rule was utilized (Equation 8). Alternatively, the actions could be explicitly added in the generative model $p(\mathbf{o}, \boldsymbol{s}, \mathbf{a})$. When doing so, this problem can be efficiently solved using factor graphs [35], [26], [27].

A key property of the AIC is the coupling between control and state-estimation. As shown, to perform any meaningful control, the state must be biased towards the target. Thus our belief about the true state is only accurate when the system reaches the target. The coupling between estimation and control makes some quantities lose their physical meaning. For instance Σ_o does not represent the uncertainty of the joint position encoders anymore. It is rather a combination of the encoder's uncertainty and how far the target is from the current position. Future work will investigate this further.

VIII. Conclusions

In this paper, we demonstrate how minimizing a single quantity: variational free-energy, effective state-estimation, control, and learning can be performed for a robotic manipulator. Online estimation of relevant quantities can be achieved using gradient descent on the free-energy for each iteration of the controller. We introduce a temporal parameter and show that when τ approaches zero, the approach converts to a PID controller and if τ approaches ∞, it converts to a filter. We then demonstrated the effectiveness of the framework for a 7 DOF robotic arm and showed adaptability and robustness ourperforming previous work. We showed that our approach improves robustness, damps oscillations, and adapts to different payloads.

Acknowledgment

The authors thank Mees Vanderbroeck, Matias Mattamala and Charlie Street for helpful comments and feedback. This work was supported by UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN), and Offshore Robotics for Certification of Assets (ORCA) hubs [EP/R026084/1, EP/R026173/1].

References

[1] Jesús Enrique Sierra and Matilde Santos. Wind and payload disturbance rejection control based on adaptive neural estimators: application on quadrotors. *Complexity*, 2019.

[2] Bin Wei. Adaptive control design and stability analysis of robotic manipulators. In *Actuators*, volume 7, page 89. Multidisciplinary Digital Publishing Institute, 2018.

[3] Bustanul Arifin, Bhakti Yudo Suprapto, Sri Arttini Dwi Prasetyowati, and Zainuddin Nawawi. The lateral control of autonomous vehicles: A review. In *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 277–282. IEEE, 2019.

[4] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: a process theory. *Neural computation*, 29(1):1–49, 2017.

[5] Karl Friston. Friston k. the free-energy principle: a rough guide to the brain? trends cogn sci 13: 293-301. *Trends in cognitive sciences*, 13:293–301, 07 2009.

[6] Giovanni Pezzulo, Francesco Rigoli, and Karl J Friston. Hierarchical active inference: A theory of motivated control. *Trends in cognitive sciences*, 22(4):294–306, 2018.

[7] Raphael Kaplan and Karl J Friston. Planning and navigation as active inference. *Biological cybernetics*, 112(4):323–343, 2018.

[8] Christopher L Buckley, Chang Sub Kim, Simon McGregor, and Anil K Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79, 2017.

[9] Pablo Lanillos and Gordon Cheng. Active inference with function learning for robot body perception. In *International Workshop on Continual Unsupervised Sensorimotor Learning, IEEE Developmental Learning and Epigenetic Robotics (ICDL-Epirob)*, 2018.

[10] Guillermo Oliver, Pablo Lanillos, and Gordon Cheng. Active inference body perception and action for humanoid robots. *arXiv preprint arXiv:1906.03022*, 2019.

[11] Corrado Pezzato, Riccardo MG Ferrari, and Carlos Hernandez. A novel adaptive controller for robot manipulators based on active inference. *IEEE Robotics and Automation Letters*, 2020.

[12] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[13] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[14] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

[15] Marie Dillon Dahleh and William T Thomson. Theory of vibration with applications. *Prentice-Hall Inc*, 1998.

[16] Léo Pio-Lopez, Ange Nizard, Karl Friston, and Giovanni Pezzulo. Active inference and robot control: a case study. *Journal of The Royal Society Interface*, 13(122):20160616, 2016.

[17] 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)*, pages 4083–4090. IEEE, 2018.

[18] Cansu Sancaktar and Pablo Lanillos. End-to-end pixel-based deep active inference for body perception and action. *arXiv preprint arXiv:2001.05847*, 2019.

[19] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[20] RG Walters and MM Bayoumi. Application of a self-tuning pole-placement regulator to an industrial manipulator. In *1982 21st IEEE Conference on Decision and Control*, pages 323–329. IEEE, 1982.

[21] 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 *1st International Workshop on Active Inference, ECML/PKDD 2020*, Ghent, Belgium, 2020.

[22] Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229 – 252, 2008.

[23] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. Robotic fault detection and fault tolerance: A survey. *Reliability Engineering and System Safety*, 46:139–158, 1994.

[24] A. Meera and M Wisse. Free energy principle based state and input observer design for linear systems with colored noise. In *2020 American Control Conference (ACC)*, pages 5052–5058, 2020.

[25] Thijs W van de Laar and Bert de Vries. Simulating active inference processes by message passing. *Frontiers in Robotics and AI*, 6(20), 2019.

[26] Thijs van de Laar, Ayça Özçelikkale, and Henk Wymeersch. Application of the free energy principle to estimation and control. *arXiv preprint arXiv:1910.09823*, 2019.

[27] Mees Vanderbroeck, Mohamed Baioumy, Daan van der Lans, Rens de Rooij, and Tiis van der Werf. Active inference for robot control: A factor graph approach. *Student Undergraduate Research E-journal!*, 5:1–5, 2019.

[28] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

[29] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[30] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[31] Karl Friston. Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211, 2008.

[32] Mohamed Baioumy, Matias Mattamala, Paul Duckworth, Bruno Lacerda, and Nick Hawes. Adaptive manipulator control using active inference with precision learning. In *UKRAS20 Conference: "Robots into the real world" Proceedings*, Lincoln, United Kingdom, May 2020.

[33] Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211, 2017.

[34] Mohamed Baioumy, Matias Mattamala, and Nick Hawes. Variational inference for predictive and reactive controllers, 2020.

[35] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007.

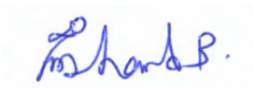
Statement of Authorship for joint/multi-authored papers for PGR thesis

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

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

Title of Paper	Active Inference for Integrated State-Estimation, Control, and Learning
Publication Status	Published
Publication Details	Mohamed Baioumy, Paul Duckworth, Bruno Lacerda and Nick Hawes, "Active Inference for Integrated State-Estimation, Control, and Learning", IEEE Intl. Conf. on Robotics and Automation (ICRA), 2021

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	I led the development of the idea in collaboration with Bruno Lacerda and Nick Hawes. I performed the literature review with assistance from Paul Duckworth. I implemented the algorithm and all the experiments. The paper was written by me with support from Paul Duckworth, Bruno Lacerda, and Nick Hawes.		
Signature		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement		
Signature 	Date	13/12/2023

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

B. ***Unbiased active inference for classical control***

In this paper, we address the limitations of the active inference controller (AIC) in robotic applications by introducing the unbiased active inference controller (u-AIC). Our analysis identifies critical issues with the AIC, such as biased state estimation and implicit control models. The proposed u-AIC overcomes these drawbacks while retaining the AIC's advantages. Our approach is validated through simulations on a 2-DOF arm and real-world experiments on a 7-DOF manipulator. The contributions of this work are two-fold: first, providing a comprehensive analysis of the AIC's limitations in robot control; second, the formalization of the u-AIC with theoretical convergence proofs and controller extensions, improving its applicability and efficacy in practical robotic systems.

This paper is concerned with state-estimation and control (S&C), it deals with continuous random variables and it's validated on a real robot.

Now we discuss the potential utility of utilizing probabilistic inference in this paper.

- **Novel Capabilities (×):** This paper does not directly introduce a novel capability compared to state-of-the-art control. The presented controller (u-AIC) has capabilities outperforming the active inference controller (AIC) discussed in previous work but isn't compared to any other control methods. Follow-up work however (e.g. Section IV-B) shows novel capabilities possible when using the u-AIC.
- **Coherence (✓):** This paper performs state-estimation and control in a coherent manner. This benefits the researcher with simplicity of notation, derivation of equations and analyzing results. This also makes insights about how goal states are embedded in the controller clear to see.
- **Joint Optimization (✓) :** In this paper, control and state-estimation are jointly optimized by minimizing free-energy. This results in simplicity and computational efficiency.
- **Automated Solvers (✓):** This paper uses a standard solver to achieve both state-estimation and control.
- **Connection to Literature (✓) :** This paper connects active inference (a method from computational neuroscience) to control theory principles. We particularly highlight the idea of biased goal states present in active inference and how it can hinder control in robotics.

The paper starts on the following page.

Unbiased Active Inference for Classical Control

Mohamed Baioumy*[1], Corrado Pezzato*[2], Riccardo Ferrari[3], Nick Hawes[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`.**

Fig. 1: Manipulation of delicate items in a human-shared store environment.

I. Introduction

Active inference is a mathematical framework prominent in computational neuroscience [1]. 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 [2]. 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 present an analysis of 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 [3] solely for fault-tolerant control. The effort in understanding the limits of the AIC for robot control started with our previous work [3], 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 [4] but also influences precision learning (learning the precision/covariance of the sensory model) making the model parameters converge to a biased value [4], [5]. 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 [6], [3]. 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 [3] 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, originally introduced in [3]. We demonstrate the properties of the u-AIC in Section IV, as well as the convergence of both the state estimation and control which is still missing for the AIC [7].

The *contributions* of this paper are twofold: 1) we con-

* Authors with equal contribution

[1]Mohamed Baioumy and Nick Hawes are with the Oxford Robotics Institute, Oxford University, `[mohamed, nickh]@robots.ox.ax.uk`

[2]Corrado Pezzato is with the Cognitive Robotics Department, TU Delft, `c.pezzato@tudelft.nl`

[3]Riccardo Ferrari is with the Department of Systems and Control, TU Delft, `r.ferrari@tudelft.nl`

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

cretely 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 that want to apply it for robot control, such that better and safer systems can be built using this framework.

A. *Related work*

Active inference has been proposed in neuroscience as a general theory of the brain [8] and was concisely reported in [9] using a notation and language closer to engineering and control. We refer to the active inference controller in [9] as AIC, from which the first real-world applications for robot control [7], [10] were derived. Recently, active inference has been applied for robot control on a broader variety of settings and tasks. This includes work on robust state-estimation [11], [12], adaptive control [7], [4], fault-tolerant control [3], [13], reinforcement learning [14], planning under uncertainty [15], [16], human-robot interaction [17], [18] and more. The recent survey in [2] 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 [19], LQG control [20] and model-predictive control [21].

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. Work in [22], [4] has shown the relationship between the AIC and PID controllers while the relationship between the AIC, LQR control, and Kalman filters is discussed in [23], [20]. Finally, in [6], [3], some limitations of the AIC were discussed limited to fault-tolerant control, which motivated the introduction of the u-AIC [3]. 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.

B. *Structure of the paper*

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

II. PRELIMINARIES

In this section, we concisely report the AIC formulation as used in previous work (e.g. [7], [4], [24], [10]), which specified the AIC for robot control according to the theory in [9]. In this section, we only report the crucial equations to understand the limitations of the AIC in Section III, an interested reader is referred to [7] for all the details.

A. *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:

$$p(\boldsymbol{x}, \boldsymbol{y}) = \underbrace{p(\boldsymbol{y}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x}).}_{prior} \tag{1}$$

A visual representation of this model is presented in fig. 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 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. (1) are Gaussian, the free energy becomes a sum of least square terms [7].

B. *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 [7]:

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

The free energy is a weighted sum of prediction errors up to a constant K resulting from the derivations [9]. 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 [7], [10], [3] 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 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, [7], the AIC will make the system behave like a first-order linear system with desired goal position $\boldsymbol{\mu}_g$. By definition [9], [7], it holds:

$$\boldsymbol{g}^{(i)} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)},\ \boldsymbol{f}^{(i)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^{(i)},\ \boldsymbol{g}^{(0)} = \boldsymbol{g},\ \boldsymbol{f}^{(0)} = \boldsymbol{f}. \tag{3}$$

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

C. State estimation

State estimation in the AIC is achieved by gradient descent on the free-energy [9], [25], [7] with the update rule:

$$\dot{\tilde{\boldsymbol{\mu}}} = \frac{d}{dt}\tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{\mu}}}, \tag{4}$$

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

D. Control

In the AIC, the control input $\boldsymbol{u}$ is also computed through gradient descent on $\mathcal{F}$. As can be seen by eq. (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 [9], [7]:

$$\dot{\boldsymbol{u}} = -\kappa_a \frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{u}} \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{y}}} \tag{5}$$

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 [7], [10], [3], [4] 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:

$$\dot{\boldsymbol{u}} \approx -\kappa_a \frac{\partial \mathcal{F}}{\partial \tilde{\boldsymbol{y}}} = -\kappa_a \sum_y \frac{\partial \mathcal{F}}{\partial \boldsymbol{y}}. \tag{6}$$

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:

$$\dot{\boldsymbol{u}} = -\kappa_a \left[\Sigma_{y_q}^{-1}(\boldsymbol{y}_q - \boldsymbol{\mu}) + \Sigma_{y_{\dot{q}}}^{-1}(\boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}')\right]. \tag{7}$$

III. 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 III-A 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 III-B 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. (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.

A. Limitation #1: biased state estimation

Before formally analyzing the controller, we provide a visual explanation using factor graphs. In fig. 2, the generative model of the AIC in eq. (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 now we can also compute a control action that moves the agent to the goal (using eq. (6)). However, the drawback is that state estimation is inaccurate.

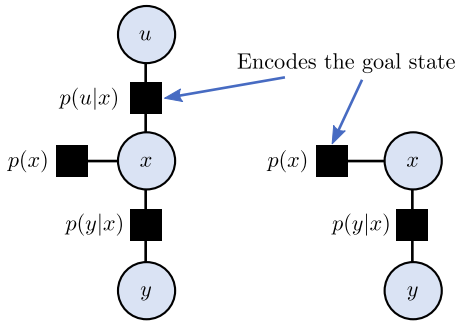


Fig. 2: Illustration of the u-AIC (left) and the AIC (right) for a one-dimensional problem. Circles indicate random variables. Black boxes indicate probability distributions.

1) 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. (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:

$$\mathcal{F} = \frac{1}{2}\left[\frac{(y - g(\mu))^2}{\sigma_y} + \frac{(\mu' - f(\mu))^2}{\sigma_\mu}\right] + K \tag{8}$$

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 $g(\mu) = \mu$ [7]. At steady state, it holds that

$$\frac{\partial \mathcal{F}}{\partial \mu} = -\frac{(y - \mu)}{\sigma_y} + \frac{(\mu' - \mu_g + \mu)}{\sigma_\mu} = 0. \tag{9}$$

Additionally, when solving for μ we can show that

$$\mu = \frac{\sigma_\mu y + \sigma_y \mu_g - \sigma_y \mu'}{\sigma_\mu + \sigma_y}. \tag{10}$$

From eq. (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 VI-A), 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.

2) ***Biased precision learning****:* Past work on robot control such as [7], [10] 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 [26], [27].

There exists a body of work related to active inference which explores precision learning., e.g. methods such as Dynamic Expectation Maximization and generalized filtering [28], [29]. 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. Work in [5], [4] shows 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. (8). The constant K can be expanded to include the variances as (see [9]):

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

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

$$\frac{\partial \mathcal{F}}{\partial \sigma_y} = -\frac{(y-\mu)^2}{2\sigma_y^2} + \frac{1}{2\sigma_y}. \tag{11}$$

Setting this expression to zero, we obtain:

$$\sigma_y = (y-\mu)^2. \tag{12}$$

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 [21].

3) ***Limitation in fault-tolerant control****:* The AIC was used in [6] for fault tolerant control, 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 [3], where a first version of the u-AIC has been proposed. We refer the reader to [3] for additional details.

B. Limitation #2: implicit modelling of actions

The control law for the AIC is computed using gradient descent on $\mathcal{F}$, as in eq. (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 [5], [7]. 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. (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. (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, under the assumptions explained in [4] and [5], 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. (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 results 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 VI 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. In case one has information about the system, designing a feed-forward signal to supplement the feedback controller consistently improves performance.

IV. UNBIASED ACTIVE INFERENCE CONTROLLER

A first version of the u-AIC has been introduced in [3] 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.

A. *Derivation of the u-AIC*

In this section, we describe the u-AIC as introduced in [3], 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 modelled explicitly:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \quad \underbrace{p(\boldsymbol{y}|\boldsymbol{x})}_{observation\ model} \quad \underbrace{p(\boldsymbol{x})}_{prior} \tag{13}$$

A visual representation of the u-AIC can be seen in fig. 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 [4], 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 $\boldsymbol{\mu}_x$ [30]. 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 [3]:

$$\mathcal{F} = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}) + C \tag{14}$$

Considering eq. (13) and assuming Gaussian distributions, $\mathcal{F}$ becomes:

$$\begin{aligned}\mathcal{F} = \frac{1}{2}(&\boldsymbol{\varepsilon}_y^\top \Sigma_y^{-1} \boldsymbol{\varepsilon}_y + \boldsymbol{\varepsilon}_x^\top \Sigma_x^{-1} \boldsymbol{\varepsilon}_x \\ &+ \boldsymbol{\varepsilon}_u^\top \Sigma_u^{-1} \boldsymbol{\varepsilon}_u + \ln|\Sigma_u \Sigma_y \Sigma_x|) + C,\end{aligned} \tag{15}$$

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 propagating forward in time the current state belief using the following simplified discrete time model:

$$\hat{\boldsymbol{x}}_{k+1} = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \boldsymbol{\mu}_{x,k} \tag{16}$$

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:

$$\dot{\boldsymbol{\mu}}_u = -\kappa_u \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_{\boldsymbol{u}}}, \quad \dot{\boldsymbol{\mu}}_x = -\kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_{\boldsymbol{x}}}, \tag{17}$$

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

$$\frac{\partial F}{\partial \boldsymbol{\mu}_u} = \Sigma_u^{-1}(\boldsymbol{\mu}_u - \boldsymbol{f}^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_g)) \tag{18}$$

B. *Proof of convergence*

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

$$\mathcal{F} = \frac{1}{2}\left[\frac{(y-\mu_x)^2}{\sigma_y} + \frac{(\mu_u - f^*(\cdot))^2}{\sigma_u} + \frac{(\mu_x - \hat{x})^2}{\sigma_x}\right] + K \tag{19}$$

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

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_u} = 0, \quad \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_x} = 0, \tag{20}$$

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

$$\mu_x = \frac{\sigma_x y + \sigma_y \hat{x}}{\sigma_x + \sigma_y}. \tag{21}$$

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. (10).

C. *Extensions of the u-AIC for control*

The u-AIC can be extended for richer control by modifying the probabilistic model in eq. (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.

*1) **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 be also added. In that case, the generative model would be:

$$\frac{1}{\alpha} p(\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) \tag{22}$$

where α is a normalization constant. Note that, this denominator does not need to be computed explicitly. To

achieve state estimation and control, we simply need to minimize the free energy (which maximizes the likelihood of the model). We do not need to exactly compute the free energy or the likelihood.

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:

$$\mu_u = \frac{f_{ol}(\mu_x)\sigma_u + f^*(\mu_g, \mu_x)\sigma_{ol}}{\sigma_{ol} + \sigma_u}. \tag{23}$$

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.

*2) **Adding control costs**:* Alternative to adding 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.

*3) **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) [31].

A comparison of the standard AIC and u-AIC against MPC and impedance control can be found in [24]. Future work could address the comparison of the proposed extensions of the u-AIC against other classical controllers.

V. Relationship between the AIC and u-AIC

A. 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 to move towards the target. This results in a biased state estimate as seen in eq. (10). In contrast, the u-AIC has an unbiased belief over the state as seen in eq. (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. 2).

B. Control law

In the u-AIC, actions are explicitly modelled 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. (5)). The eventual control law of the AIC is also directly dependent on the measurement and the number of measurements eq. (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. (21) and allows us to encode control costs, smoothing effects and feed-forward control signals (see eq. (23)).

C. 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. 3. A discussion on the role of modularity in AIC and the general control architecture can be found in [32]. The u-AIC on the other hand has a more traditional flow

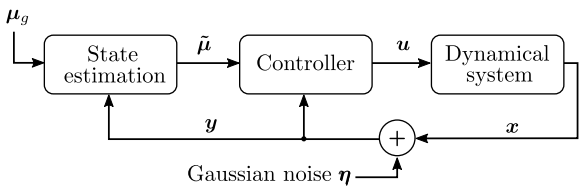


Fig. 3: Control diagram of the AIC

of 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.

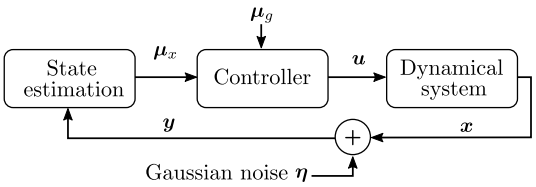


Fig. 4: Control diagram of the u-AIC.

VI. Experimental evaluation

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

A. Simulation

The simulation scenario is depicted in fig. 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.

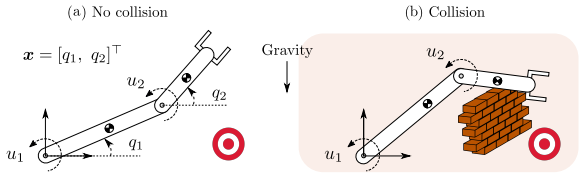


Fig. 5: Scenarios considered to illustrate the incorrect state estimation due to the bias towards the target (red dot). In (b) an obstacle occludes the way and blocks the arm.

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. 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. 6.

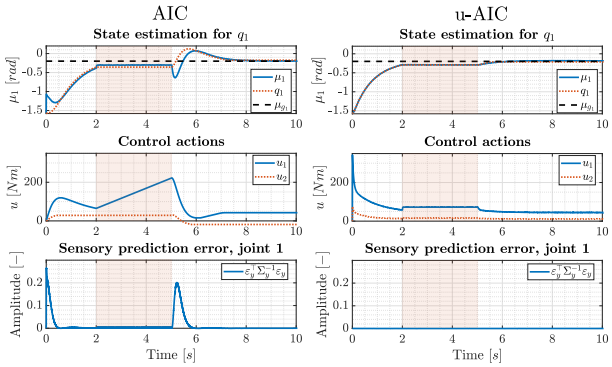


Fig. 6: Simulation results of a 2-DOF robot arm collision (orange area) for $t_c = 3s$. For readability, we only report the first joint since the behavior is similar for the second one.

The collision scenario in fig. 5 highlights a few limitations:

*1) **Incorrect state estimation**:* Let us consider the first row of the plots in fig. 6. When the AIC is not able to 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. (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 $\in [1, 3]s$ at time $\in [0, 3]s$. Table I 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,

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

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 as described in detail in [3]. The sensory prediction errors for the AIC can in fact increase also due to collisions with the environment (see the last row of plots in fig. 6). The u-AIC is instead insensitive.

*2) **Monotonic increase of control input**:* The second row of the plots in fig. 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. (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.

B. 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].

*1) **Reference tracking**:* The performance in terms of reference tracking of a sinusoidal wave are reported in fig. 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.

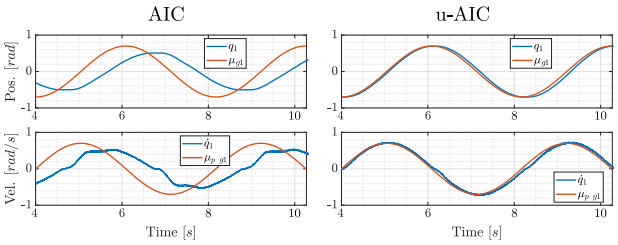


Fig. 7: Experiments with real Panda arm on reference tracking. For readability, we only report the first joint.

The AIC performs poorly in terms of tracking errors. This is due to three main reasons: 1) the AIC as defined in [9], [7] only allows for position reference through its generative

[1] https://www.youtube.com/watch?v=jI-zX8XvfgI

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.

*2) **Collision behavior**:* The collision behavior is reported in fig. 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.

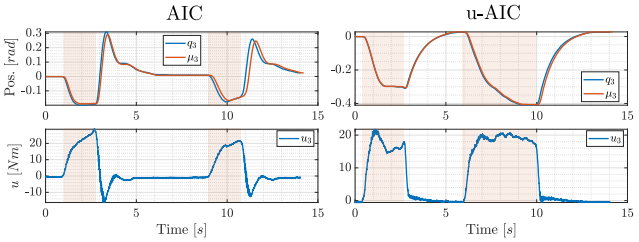


Fig. 8: Experiments with real Panda arm during unwanted interaction (orange area). For readability we only report the third joint, being the most affected one. Faster convergence to zero steady-state error for the u-AIC can be achieved through tuning of the integral action.

VII. 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 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 in the real world with a 7-DOF robot manipulator.

References

[1] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: a process theory," *Neural computation*, vol. 29, no. 1, pp. 1–49, 2017.

[2] P. Lanillos, C. Meo, C. Pezzato, A. A. Meera, M. Baioumy, W. Ohata, A. Tschantz, B. Millidge, M. Wisse, C. L. Buckley, *et al.*, "Active inference in robotics and artificial agents: Survey and challenges," *arXiv preprint arXiv:2112.01871*, 2021.

[3] M. Baioumy, C. Pezzato, R. Ferrari, C. H. Corbato, and N. Hawes, "Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference," in *European Control Conf. (ECC)*, 2021.

[4] M. Baioumy, P. Duckworth, B. Lacerda, and N. Hawes, "Active inference for integrated state-estimation, control, and learning," in *Proc of IEEE Int. Conf. on robotics and automation (ICRA)*, 2021.

[5] M. Baltieri and C. L. Buckley, "PID control as a process of active inference with linear generative models," *Entropy*, vol. 21, no. 3, 2019.

[6] C. Pezzato, M. Baioumy, C. H. Corbato, N. Hawes, M. Wisse, and R. Ferrari, "Active inference for fault tolerant control of robot manipulators with sensory faults," in *1st Int. Workshop on Active Inference, ECML PKDD*, ser. Communications in Computer and Information Science, Springer, Ed., vol. 1326, 2020.

[7] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, 2020.

[8] K. J. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11(2), pp. 27–138, 2010.

[9] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.

[10] G. Oliver, P. Lanillos, and G. Cheng, "An empirical study of active inference on a humanoid robot," *IEEE Transactions on Cognitive and Developmental Systems*, 2021.

[11] P. Lanillos and G. Cheng, "Adaptive robot body learning and estimation through predictive coding," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[12] A. Meera and M. Wisse, "Free energy principle based state and input observer design for linear systems with colored noise," in *2020 American Control Conf. (ACC)*, 2020, pp. 5052–5058.

[13] M. Baioumy, C. Pezzato, C. H. Corbato, N. Hawes, and R. Ferrari, "Towards stochastic fault-tolerant control using precision learning and active inference," in *IWAI*. Springer, 2021.

[14] A. Tschantz, B. Millidge, A. K. Seth, and C. L. Buckley, "Reinforcement learning through active inference," *arXiv preprint arXiv:2002.12636*, 2020.

[15] L. Da Costa, N. Sajid, T. Parr, K. Friston, and R. Smith, "The relationship between dynamic programming and active inference: The discrete, finite-horizon case," *arXiv preprint arXiv:2009.08111*, 2020.

[16] M. Baioumy, P. Duckworth, B. Lacerda, and N. Hawes, "On solving a stochastic shortest-path markov decision process as probabilistic inference," in *IWAI*. Springer, 2021.

[17] H. F. Chame, A. Ahmadi, and J. Tani, "A hybrid human-neurorobotics approach to primary intersubjectivity via active inference," *Frontiers in Psychology*, vol. 11, p. 3207, 2020.

[18] W. Ohata and J. 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," *Frontiers in Neurorobotics*, vol. 14, Sep 2020.

[19] T. van de Laar, İ. Şenöz, A. Özçelikkale, and H. Wymeersch, "Chance-constrained active inference," *Neural Computation*, 2021.

[20] T. van de Laar, A. Özçelikkale, and H. Wymeersch, "Application of the free energy principle to estimation and control," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4234–4244, 2021.

[21] M. Baioumy, M. Mattamala, and N. Hawes, "Variational inference for predictive and reactive controllers," in *BAIN-PIL workshop, ICRA*, Paris, France, 2020.

[22] M. Baltieri and C. L. Buckley, "A probabilistic interpretation of PID controllers using active inference," in *Int. Conf. on Simulation of Adaptive Behavior*. Springer, 2018, pp. 15–26.

[23] ——, "On kalman-bucy filters, linear quadratic control and active inference," *arXiv preprint arXiv:2005.06269*, 2020.

[24] C. Meo and P. Lanillos, "Multimodal vae active inference controller," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[25] K. Friston, K. Stephan, B. Li, and J. Daunizeau, "Generalised filtering," *Mathematical Problems in Engineering*, 2010.

[26] W. Vega-Brown and N. Roy, "Cello-em: Adaptive sensor models without ground truth," in *2013 IEEE/RSJ International Conf. on Intelligent Robots and Systems*. IEEE, 2013, pp. 1907–1914.

[27] T. Pfeifer, S. Lange, and P. Protzel, "Dynamic covariance estimation—a parameter free approach to robust sensor fusion," in *2017 IEEE International Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2017, pp. 359–365.

[28] K. J. Friston, N. Trujillo-Barreto, and J. Daunizeau, "Dem: a variational treatment of dynamic systems," *Neuroimage*, 2008.

[29] K. Friston, K. Stephan, B. Li, and J. Daunizeau, "Generalised filtering," *Mathematical Problems in Engineering*, vol. 2010, 2010.

[30] K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the Laplace approximation," *Neuroimage*, vol. 34(1), pp. 220–234, 2007.

[31] L. E. Olivier and I. K. Craig, "Fault-tolerant nonlinear mpc using particle filtering," *IFAC-PapersOnLine*, 2016.

[32] M. Baltieri and C. L. Buckley, "The modularity of action and perception revisited using control theory and active inference," *arXiv preprint arXiv:1806.02649*, 2018.

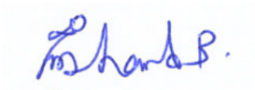
Statement of Authorship for joint/multi-authored papers for PGR thesis

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

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

Title of Paper	Unbiased Active Inference for Classical Control
Publication Status	Published
Publication Details	Mohamed Baioumy, Corrado Pezzato, Riccardo Ferrari and Nick Hawes, "Unbiased Active Inference for Classical Control", IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), 2022

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	The idea for the paper was developed by Corrado Pezzato and me, with equal contributions. I led the work on the mathematical formulation of the u-AIC (the novel algorithm). Corrado led the implementation of the algorithm and carried out the experiments. I co-led the writing of the manuscript with Corrado Pezzato, in collaboration with Riccardo Ferrari and Nick Hawes.		
Signature		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

IV. Fault detection and fault-tolerant control

Fault-tolerant control (FTC) encompasses methodologies designed to maintain a system's control performance when faults arise. Its primary objective is to ensure a system's safety and operability, achieving performance goals despite component malfunctions. This includes sensors and actuators in the case of robot manipulators. Our work focuses on sensor fault detection.

Fault detection, the initial step in FTC, involves continuously monitoring the system to identify any anomalies indicating a fault. Various techniques, from model-based methods predicting system behavior to data-driven methods analyzing operational data, facilitate this detection. Rapid and precise fault detection is vital, as early identification enables swift corrective actions, thereby minimizing potential damage or system downtime.

Upon detecting a fault, the focus shifts to fault recovery. This stage ensures the system remains operational post-detection. Recovery methods range from employing redundancy, where backup components substitute faulty ones, to reconfiguration, adjusting the control system to circumvent the malfunctioning component. Adaptive control, another approach, tweaks controller parameters based on fault data. The essence of FTC lies not just in fault detection but in guaranteeing continued safe and efficient system operations once a fault emerges.

In the following, we discuss applying probabilistic inference methods to model FTC. In Section IV-A, we present the first attempt in applying active inference to FTC and show clear limitations. Despite the limitations, we show that there is a large amount of redundancy in computation between state estimation and fault detection. This takes us to apply unbiased active inference to FTC as seen in Section IV-B. This proves to be a powerful method. Compared to the benchmark, it is computationally more efficient, it can be combined with other methods and results in better performance on the control task.

Furthermore, in Section IV-C we use an approach based on probabilistic inference, that does not utilize the active inference framework. We show how this model outperforms other FTC methods in both computational cost and performance.

A. ***Active inference for fault tolerant control of robot manipulators with sensory faults.***

In this study, we introduce a novel fault-tolerant control scheme for robotic manipulators using the active inference framework. Our approach simplifies the process of fault detection and isolation by utilizing sensory prediction errors in free-energy calculations. Validated on a 2DOF robot arm, this method can be effective in detecting and recovering from sensory faults while reducing overall system complexity. We also highlight key limitations of this approach.

Now we discuss the potential utility of utilizing probabilistic inference in this paper

- **Novel Capabilities (✓):** This paper introduces an approach based on active inference for FTC. Compared to existing approaches in the literature it shows how fault-detection calculation can be simplified, and how fault-recovery can be simplified. This is a valuable contribution. However, it's also critical to highlight that, due to many limitations (e.g. high number of false positives) this approach wouldn't be practical in many real-world settings.
- **Coherence (×):** This paper shows a lack of coherence. The fault-detection and isolation (FDI) step is not modelled directly as probabilistic inference. This means that one has to manually prove properties for this controller. In the next chapters we show how modelling FDI as probabilistic inference yields superior strategies, due to coherence, and computations beings simplified even further.
- **Joint Optimization (✓) :** Although FDI is not modelled using probabilistic inference, joint optimization is still performed. This means that computations used in probabilistic inference (used for state-estimation) are combined with computations used for FDI. The joint optimization results in a significant improvement in computational efficiency.
- **Automated Solvers (×):** FDI was not modelled as probabilistic inference and thus a custom implementation was used in this case.
- **Connection to Literature (✓) :** This paper connects active inference to approaches in fault-tolerant control. The similarity between the computations for sensory prediction errors (in active inference) and residual signals (in FTC) is highlighted and exploited for computation gain.

The paper starts on the following page.

Active Inference for Fault Tolerant Control of Robot Manipulators with Sensory Faults

Corrado Pezzato[1], Mohamed Baioumy[3], Carlos Hernández Corbato[1], Nick Hawes[3], Martijn Wisse[1], and Riccardo Ferrari[2]

[1] Cognitive Robotics, Delft University of Technology
[2] Delft Center for Systems and Control, Delft University of Technology
{c.pezzato, c.h.corbato, r.ferrari, m.wisse}@tudelft.nl
[3] Oxford Robotics Institute, University of Oxford
{mohamed, nickh}@robots.ox.ac.uk

Abstract. We present a fault tolerant control scheme for robot manipulators based on active inference. The proposed solution makes use of the sensory prediction errors in the free-energy to simplify the residuals and thresholds generation for fault detection and isolation and does not require additional controllers for fault recovery. Results validating the benefits in a simulated 2DOF manipulator are presented and the limitations of the current approach are highlighted.

Keywords: Fault-tolerant control · fault recovery · active inference · free-energy · robot manipulator

1 Introduction

Developing fault tolerant (FT) control schemes is of vital importance to bring robots outside controlled laboratories. The area of fault tolerant control has become increasingly more important in recent years, and several methods have been developed in different fields. An extensive bibliographical review and classification of FT methods can be found in [25]. Model-based FT techniques are amongst the most promising approaches [8]. 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 [17]. The two main challenges to design FT schemes are the definition of residuals and thresholds, and the design of a fault specific recovery strategy. We present a novel FT scheme for sensory faults [19, 23] based on an active inference controller (AIC) [20], which is inspired by the active inference framework. Active inference is prominent in the neuroscientific literature as a general theory of the brain [9–11] and several recent approaches in robotics have taken inspiration from it [1–3, 15, 16, 18, 20–22, 24]. In this work we investigate the utility of the active inference framework for fault-tolerant control with sensory faults. In the presented scheme, we exploit the properties of the framework to simplify the definition of both residuals and thresholds, and we provide a simple and general mechanism for sensory fault recovery. Our approach is validated on a simulated 2DOF manipulator.

2 Problem statement

We consider a robot controlled in joint space with torque commands, using an active inference controller (AIC) [20, 1]. In the following, we highlight the necessary elements and assumptions to derive an expression for the free-energy of the system, and the equations for state estimation and control. This study considers a 2-DOF robot arm, equipped with a vision system to retrieve the end effector Cartesian position $\boldsymbol{y}_v = [y_{v_x},\ y_{v_z}]^\top$, and with position and velocity sensors $\boldsymbol{y}_q,\ \boldsymbol{y}_{\dot{q}} \in \mathbb{R}^2$ for the two joints. Thus, we define $\boldsymbol{y} = [\boldsymbol{y}_q,\ \boldsymbol{y}_{\dot{q}},\ \boldsymbol{y}_v]$. The

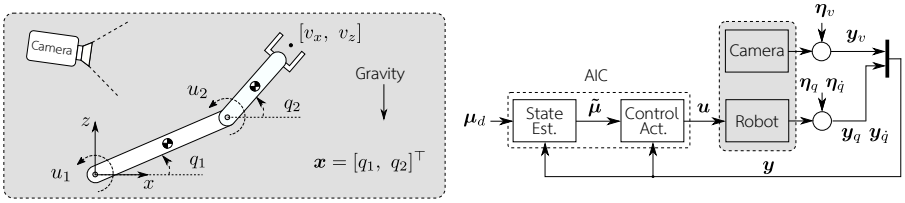


Fig. 1. 2-DOF robot arm and general AIC control scheme.

proprioceptive sensors and the camera are affected by zero mean Gaussian noise $\boldsymbol{\eta} = [\boldsymbol{\eta}_q,\ \boldsymbol{\eta}_{\dot{q}},\ \boldsymbol{\eta}_v]$. Additionally, the camera is affected by barrel distortion. The states $\boldsymbol{x}$ to be controlled are set as the joint positions $\boldsymbol{q}$ of the robot arm. We define the generative model of the state dynamics $\boldsymbol{f}(\cdot)$ 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.

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

The relation between $\boldsymbol{\mu}$ and $\boldsymbol{y}$ is expressed through the generative model of the sensory input $\boldsymbol{g} = [\boldsymbol{g}_q,\ \boldsymbol{g}_{\dot{q}},\ \boldsymbol{g}_v]$. Since we set $\boldsymbol{x} = [q_1,\ q_2]^\top$ and we can directly measure joint positions, $\boldsymbol{g}_q$ and $\boldsymbol{g}_{\dot{q}}$ and their partial derivatives are [20, 7]:

$$\boldsymbol{g}_q(\boldsymbol{\mu}) = \boldsymbol{\mu}, \quad \boldsymbol{g}_{\dot{q}}(\boldsymbol{\mu}) = \boldsymbol{\mu}', \quad \partial \boldsymbol{g}_q(\boldsymbol{\mu})/\partial \boldsymbol{\mu} = 1, \quad \partial \boldsymbol{g}_{\dot{q}}(\boldsymbol{\mu})/\partial \boldsymbol{\mu}' = 1 \tag{2}$$

Note that $\boldsymbol{\mu}'$ is the first order generalised motion of $\boldsymbol{\mu}$. To define $\boldsymbol{g}_v(\boldsymbol{\mu})$, instead, we use a *Gaussian Process Regression* (GPR) as in [16]. The training data is composed by a set of observations of the 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:

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

where $\boldsymbol{y}_{q_i},\ \boldsymbol{y}_{q_j} \in \bar{\boldsymbol{y}}_q$, and d_{ij} is the Kronecker delta function. Λ is a diagonal matrix of hyperparameters to be optimised. It holds then:

$$\boldsymbol{g}_v(\boldsymbol{y}_{\boldsymbol{q}_*}) = \begin{bmatrix} K_* K^{-1} \bar{\boldsymbol{y}}_{v_x} \\ K_* K^{-1} \bar{\boldsymbol{y}}_{v_z} \end{bmatrix} \quad \boldsymbol{g}_v(\boldsymbol{y}_{\boldsymbol{q}_*})' = \begin{bmatrix} -\Lambda^{-1}(\boldsymbol{y}_{\boldsymbol{q}_*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{\boldsymbol{q}_*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_x] \\ -\Lambda^{-1}(\boldsymbol{y}_{\boldsymbol{q}_*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{\boldsymbol{q}_*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_z] \end{bmatrix} \tag{3}$$

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

Given the generative models $\boldsymbol{f}$ and $\boldsymbol{g}$ as before, we can define an expression for the free-energy $\mathcal{F}$. Under Laplace approximation, and considering normally distributed uncorrelated noise and generalised motions up to second order, the free-energy for the 2-DOF robot arm can be expressed as:

$$\mathcal{F} = \frac{1}{2}\sum_i \left(\boldsymbol{\varepsilon}_i^\top P_i \boldsymbol{\varepsilon}_i + \ln|P_i|\right) + C, \qquad i \in \{y_q,\ y_{\dot{q}},\ y_v,\ \mu,\ \mu'\} \tag{4}$$

where and C is a constant and P_i defines a precision (or inverse covariance) matrix. Note that we set $\tau = 1$ as in [20]. The terms $\boldsymbol{\varepsilon}_i = (\boldsymbol{y}_i - \boldsymbol{g}_i(\mu))$ 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. The model prediction errors are instead defined considering the desired state dynamics as $\boldsymbol{\varepsilon}_\mu = (\boldsymbol{\mu}' - \boldsymbol{f}(\mu))$ and $\boldsymbol{\varepsilon}_{\mu'} = (\boldsymbol{\mu}'' - {}^{\partial \boldsymbol{f}(\mu)}/_{\partial \boldsymbol{\mu}\mu'})$. In particular, for the 2-DOF example it results that $\boldsymbol{\varepsilon}_q = (\boldsymbol{y}_q - \boldsymbol{\mu})$, $\boldsymbol{\varepsilon}_{\dot{q}} = (\boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}')$, $\boldsymbol{\varepsilon}_v = (\boldsymbol{y}_v - \boldsymbol{g_v}(\boldsymbol{\mu}))$, 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 (4), an interested reader can refer to [20, 6, 7].

Finally, one can compute the generalised state estimates $\boldsymbol{\mu}$, $\boldsymbol{\mu}'$, and $\boldsymbol{\mu}''$, and control actions $\boldsymbol{u}$ by minimizing $\mathcal{F}$ through gradient descent [11]:

$$\dot{\boldsymbol{\mu}} = \boldsymbol{\mu}' - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}}, \quad \dot{\boldsymbol{\mu}}' = \boldsymbol{\mu}'' - \kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}'}, \quad \dot{\boldsymbol{\mu}}'' = -\kappa_\mu \frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}''} \tag{5}$$

$$\dot{\boldsymbol{u}} = -\kappa_a \frac{\partial \boldsymbol{y}_q}{\partial \boldsymbol{u}} P_{y_q}(\boldsymbol{y}_q - \boldsymbol{\mu}) - \kappa_a \frac{\partial \boldsymbol{y}_{\dot{q}}}{\partial \boldsymbol{u}} P_{y_{\dot{q}}}(\boldsymbol{y}_{\dot{q}} - \boldsymbol{\mu}') - \kappa_a \frac{\partial \boldsymbol{y}_v}{\partial \boldsymbol{u}} P_{y_v}(\boldsymbol{y}_v - \boldsymbol{g}_v(\boldsymbol{\mu})) \tag{6}$$

Note that P_{y_q}, $P_{y_{\dot{q}}}$ and P_{y_v} are the precision matrices representing the confidence about sensory inputs. The higher the confidence in a sensor, the more reliable its measurements are assumed to be. Following [18, 20], we set $\partial \boldsymbol{y}_q / \partial \boldsymbol{u}$ and $\partial \boldsymbol{y}_{\dot{q}} / \partial \boldsymbol{u}$ to the identity, approximating the true relationships with only their sign. Similar considerations can be made for the relation between commanded torques and Cartesian displacements. The sign of $\partial \boldsymbol{y}_v / \partial \boldsymbol{u}$ depends on the combination of the two joint angles. For instance, operating the end effector in the fourth quadrant with $-\pi/2 \leq q_1 \leq 0$, a positive u_1 will lead to positive increments of both x and z coordinates. More advanced methods to determine these partial derivatives are out of the scope of this work, but definitely possible and encouraged.

3 A fault tolerant scheme based on active inference

In this section we define a fault tolerant scheme as in Fig. 2, using the SPE to build residuals for fault detection. We also show how the sensory redundancy and precision matrices can be used for fault recovery, such that we do not need to generate extra signals for detection as in conventional approaches, and we simplify fault recovery.

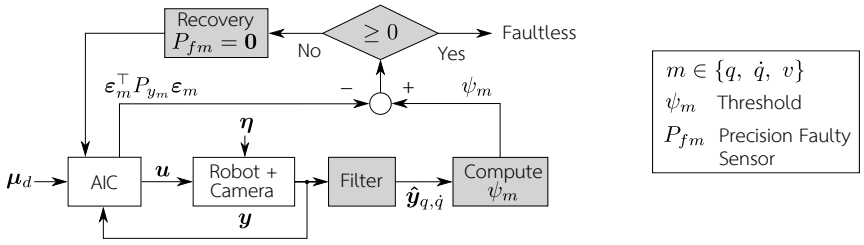


Fig. 2. AIC with additional elements (in gray) for fault tolerant control.

Threshold for fault detection and isolation (FDI) Even though the SPE can be seen as residuals for fault detection purposes, there is a substantial difference. In active inference, the belief $\boldsymbol{\mu}$ is biased towards a given goal, so the SPE can also increase for causes which are not related to sensory faults (i.e. the robot is stuck due to a collision). This precludes the use of established fault detection techniques to define a threshold ψ_m for FDI. To solve this issue we consider the quadratic form $\boldsymbol{\varepsilon}_m^\top P_{y_m} \boldsymbol{\varepsilon}_m$, where $\boldsymbol{\varepsilon}_m = \boldsymbol{y}_m - \boldsymbol{g}_m(\boldsymbol{\mu})$ with $m \in \{q,\ \dot{q},\ v\}$. The core idea is to compute an upper bound on this quadratic term. The first step is to compute an estimate for the prediction errors using the generative model as $\hat{\boldsymbol{\varepsilon}}_m = \boldsymbol{g}_m(\hat{\boldsymbol{x}}) - \boldsymbol{g}_m(\boldsymbol{\mu})$. The estimate of the joint positions $\hat{\boldsymbol{x}}$ is obtained through a filter using position and velocity measurements. Defining $\dot{\boldsymbol{x}} = \boldsymbol{z}$, we can write:

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{z}} + H_1(\boldsymbol{y}_q - \hat{\boldsymbol{x}}) \qquad \dot{\hat{\boldsymbol{z}}} = H_2(\boldsymbol{y}_{\dot{q}} - \hat{\boldsymbol{z}}) \tag{7}$$

where $H_1,\ H_2$ are diagonal positive definite matrices. The estimation error can be made arbitrary small by choosing high gains H_1 and H_2 [12].

We can represent the sensory input $\boldsymbol{y}$ as a function of the ground truth $\boldsymbol{x}$ as:

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\gamma} + \boldsymbol{\eta}\,, \tag{8}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^6$ is a vector representing the process uncertainties introduced by the generative models $\boldsymbol{g}$, and $\boldsymbol{\eta}$ is the measurement noise. The sensory prediction error for a generic sensor m can then be written as:

$$\boldsymbol{\varepsilon}_m = \hat{\boldsymbol{\varepsilon}}_m + \underbrace{\boldsymbol{g}_m(\boldsymbol{x}) - \boldsymbol{g}_m(\hat{\boldsymbol{x}}) + \boldsymbol{\gamma}_m + \boldsymbol{\eta}_m}_{\boldsymbol{\delta}_m} = \hat{\boldsymbol{\varepsilon}}_m + \boldsymbol{\delta}_m \tag{9}$$

where $\boldsymbol{\delta}_m$ is the total uncertainty including process and measurement noise. The j-th entry $\delta_m(j)$ is a scalar total uncertainty associated with a specific sensor. Since we operate the robot in a finite workspace, with specific physical limits, the states of the system $\boldsymbol{x}$, the sensory input $\boldsymbol{y}$ and the internal belief $\boldsymbol{\mu}$ remain bounded in a compact region $\mathcal{R} = \mathcal{R}_x \times \mathcal{R}_y \times \mathcal{R}_\mu \subset \mathbb{R}^2 \times \mathbb{R}^6 \times \mathbb{R}^2$, before and after the occurrence of a fault. We also suppose that the noise $\boldsymbol{\eta}$ affecting the position and velocity sensors, and the camera, can be bounded. This means $||\boldsymbol{\eta}(t)||_2 \leq \bar{\boldsymbol{\eta}}$, where $\bar{\boldsymbol{\eta}}$ is a known value. Since the AIC does not require the full dynamical model of the robot arm, the characterization of the model uncertainties $\boldsymbol{\gamma}$ due

to $\boldsymbol{g}(\cdot)$ is straightforward: $\boldsymbol{g}_q(\cdot)$ and $\boldsymbol{g}_{\dot{q}}(\cdot)$ are just an identity, so no uncertainty is introduced, while for $\boldsymbol{g}_v(\cdot)$ we can retrieve the model uncertainty from the covariance matrix of the GPR. The sensory prediction errors $\boldsymbol{\varepsilon}_m$ and $\hat{\boldsymbol{\varepsilon}}_m$ are then bounded quantities, thus we can define an upper bound for the quadratic term $\boldsymbol{\varepsilon}_m^\top P_{y_m} \boldsymbol{\varepsilon}_m$.

Definition 1. *Given a maximum uncertainty $\bar{\boldsymbol{\delta}}_m$ such that $|\delta_m(j)| \leq \bar{\delta}_m(j)\ \forall j$, we define a threshold for fault detection for sensor m as:*

$$\psi_m = \hat{\boldsymbol{\varepsilon}}_m^\top P_{y_m} \hat{\boldsymbol{\varepsilon}}_m + 2|\hat{\boldsymbol{\varepsilon}}_{y_m}^\top P_{y_m} \bar{\boldsymbol{\delta}}_m| + \bar{\boldsymbol{\delta}}_m^\top P_{y_m} \bar{\boldsymbol{\delta}}_m \tag{10}$$

Lemma 1. *In a faultless case, the quadratic form of the sensory prediction errors for a sensor m will remain below the threshold ψ_m:*

$$\boldsymbol{\varepsilon}_m^\top P_{y_m} \boldsymbol{\varepsilon}_m \leq \psi_m \tag{11}$$

Proof. Once $\bar{\boldsymbol{\delta}}_m$ is given, and since P_{y_m} is diagonal positive definite, equation (11) follows from applying the triangular inequality, considering $\boldsymbol{\varepsilon}_m$ as in equation (9).

Using *Lemma 1*, a fault in a generic sensor m is detected and isolated whenever equation (11) is violated, that is when a fault will produce an anomaly in the sensory input bigger than $\bar{\boldsymbol{\delta}}_m$. The value for the maximum uncertainty is chosen according to the standard deviation of the noise present in the sensors. Note that, in theory, a bound $\bar{\boldsymbol{\eta}}$ may not be finite or could be very large making fault detection difficult or even impossible. In practice, using multiples of the variance, we reach an acceptable compromise between false alarms and detectability. Thus, each entry of $\bar{\boldsymbol{\delta}}_m$ is set to $5\sigma_m$ where $\boldsymbol{\eta}_m \sim \mathcal{N}(0, \sigma_m^2 I)$. Doing so, the probability of having a false alarm due to the noise is less that 10^{-6}.

Fault recovery To recover from a fault we exploit the fact that the controller encodes the precision matrices P_{y_q}, $P_{y_{\dot{q}}}$ and P_{y_v}. Once a fault is detected and isolated, fault recovery can be implemented simply by setting the precision matrix of the faulty sensor to zero, that is $P_{fm} = \mathbf{0}$. This is a simple and generic mechanism to recover for any kind of sensory fault once detected.

4 Simulation results

We control the robot from the initial position $\boldsymbol{q} = [-\pi/2,\ 0]$ (rad) to the desired position $\boldsymbol{\mu}_d = [-0.6,\ 0.5]$ (rad). A fault is injected either in the encoders or in the camera during the motion of the robot, at time $t_f = 2$ (s). The maximum uncertainties are set to $\bar{\boldsymbol{\delta}}_q = [5\sigma_q,\ 5\sigma_q]^\top$ and $\bar{\boldsymbol{\delta}}_v = [5\sigma_v,\ 5\sigma_v]^\top$, where $\sigma_q = 0.001$ and $\sigma_v = 0.01$. Figure 3A reports the single normalised SPE $\varepsilon_m^\top \sigma_m^{-1} \varepsilon_m / \psi_m$, that we call $N\varepsilon_m$. Doing so, a fault is detected when the ratio is bigger than one. We assume two kinds of possible faults: 1) A fault in the encoders: the output related to the first joint freezes so $\boldsymbol{y}_q(t) = [q_1(t_f),\ q_2(t)]^\top + \boldsymbol{\eta}_q$ for $t \geq t_f$, and 2) A fault

in the camera: a misalignment is injected as bias in $\boldsymbol{y}_v$. Figure 3A shows fault detection and isolation at t_{DI}, when the normalised residual is bigger than 1. The recovered and non-recovered response of the robot in case of encoder fault is depicted in Fig. 3B. A similar response is found for camera faults.

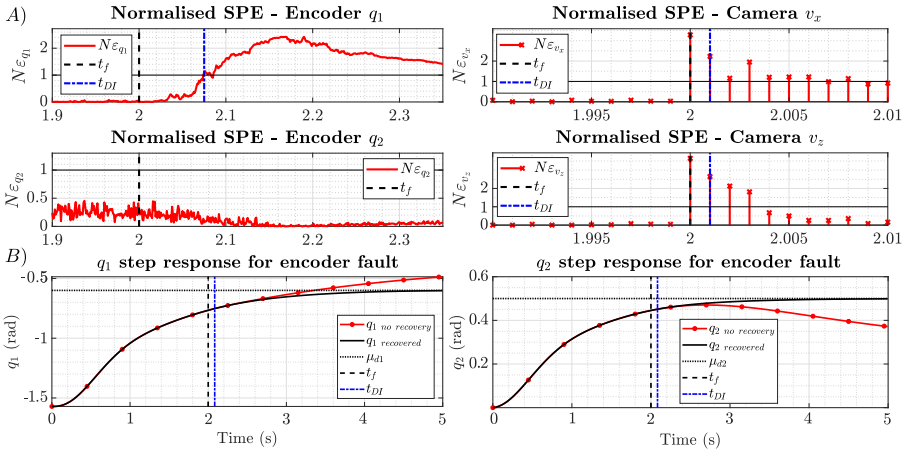


Fig. 3. A) Normalised SPE for FDI in case of encoder fault (left) or camera fault (right). B) Step response with and without recovery action in case of encoder fault.

5 Discussion and conclusion

Consider now equation (1). The time constant τ influences the generative model of the state dynamics $\boldsymbol{f}(\cdot)$, so the desired evolution of the states. As explained in [1], the AIC has two extremes depending on the value of τ^{-1} in the generative model. If $\tau^{-1} \to 0$, the estimation step has zero bias towards the target. The control action in this case will never steer the system towards the target. On the other hand, if $\tau^{-1} \to \infty$ the system is completely biased towards the target. That case is equivalent to a PID controller [1, 4, 5]. For any value in between, there is a compromise between estimation and control. The estimation and control are thus 'coupled'. This has a few limitations. First, the actions are not explicit in the model, so only sensory faults can be detected, isolated, and recovered. Second, the estimated state is always biased towards the desired state. Finally, biasing the state hinders learning model (hyper-)parameters.

What does this mean for the FT scheme presented so far? The bias could increase the SPE for reasons unrelated to sensory faults, for instance if the current target state changes to another which is further away from the current position. This prevents the direct use of the SPE as residuals in combination with established fault detection techniques, since it would cause several false positives.

It would then be beneficial to decouple estimation and control. In addition, a decoupled system could facilitate learning the hyperparameters. This could allow us to optimise the precision matrices for the SPE instead of setting P_{fm} to zero, since the precision matrices would represent the physical noise affecting the sensors. Decoupling can also help relaxing the assumption on the maximum $\bar{\boldsymbol{\delta}}_m$ which now has to be known a priori for the determination of the fault detection threshold. Approaches where the estimation and control are decoupled (similar to [3, 13, 14, 24]) for fault tolerance will be explored in future work.

To conclude, in this paper we present a novel approach for FT control based on active inference. The main novelty is the definition of an on-line threshold for FDI based on the SPE defined in the free-energy. Fault recovery is achieved by reducing the precision of faulty sensor to zero, providing a generic recovery mechanism which significantly simplifies the synthesis of reliable FT controllers. The main limitation of the proposed approach is that only sensory faults can be detected and recovered. Simulation results validated the theoretical findings. Future work will explore FT control with decoupled state-estimation and control.

References

1. Baioumy, M., Duckworth, P., Lacerda, B., Hawes, N.: Active inference for integrated state-estimation, control, and learning. arXiv preprint arXiv:2005.05894 (2020)
2. Baioumy, M., Mattamala, M., Duckworth, P., Lacerda, B., Hawes, N.: Adaptive manipulator control using active inference with precision learning. In: UKRAS (2020)
3. Baioumy, M., Mattamala, M., Hawes, N.: Variational inference for predictive and reactive controllers. In: ICRA 2020 Workshop on New advances in Brain-inspired Perception, Interaction and Learning. Paris, France (2020)
4. Baltieri, M., Buckley, C.L.: A probabilistic interpretation of pid controllers using active inference. In: International Conference on Simulation of Adaptive Behavior. pp. 15–26. Springer (2018)
5. Baltieri, M., Buckley, C.L.: Pid control as a process of active inference with linear generative models. Entropy **21**(3), 257 (2019)
6. Bogacz, R.: A tutorial on the free-energy framework for modelling perception and learning. Journal of mathematical psychology **76**, 198–211 (2017)
7. Buckley, C.L., Kim, C.S., McGregor, S., Seth, A.K.: The free energy principle for action and perception: A mathematical review. Journal of Mathematical Psychology **81**, 55–79 (2017)
8. Chen, J., Patton, R.J.: Robust model-based fault diagnosis for dynamic systems. Springer Science & Business Media, LLC (1999)
9. Friston, K.J.: The free-energy principle: a unified brain theory? Nature Reviews Neuroscience **11(2)**, 27–138 (2010)
10. Friston, K.J., Daunizeau, J., Kiebel, S.: Action and behavior: a free-energy formulation. Biological cybernetics **102(3)**, 227–260 (2010)
11. Friston, K.J., Mattout, J., Kilner, J.: Action understanding and active inference. Biological cybernetics **104(1-2)**, 137–160 (2011)

12. Khalil, H.K.: High-gain observers in nonlinear feedback control. In: 2008 International Conference on Control, Automation and Systems (2008)
13. van de Laar, T., Özçelikkale, A., Wymeersch, H.: Application of the free energy principle to estimation and control. arXiv preprint arXiv:1910.09823 (2019)
14. van de Laar, T.W., de Vries, B.: Simulating active inference processes by message passing. Frontiers in Robotics and AI **6**(20) (2019)
15. Lanillos, P., Cheng, G.: Active inference with function learning for robot body perception. In: International Workshop on Continual Unsupervised Sensorimotor Learning (ICDL-Epirob) (2018)
16. Lanillos, P., Cheng, G.: Adaptive robot body learning and estimation through predictive coding. In: IROS (2018)
17. Narendra, K.S., Balakrishnan, J.: Adaptive control using multiple models. IEEE Transaction on Automatic Control (1997)
18. Oliver, G., Lanillos, P., Cheng, G.: Active inference body perception and action for humanoid robots. arXiv preprint arXiv:1906.03022v2 (2019)
19. Paviglianiti, G., Pierri, F., Caccavale, F., Mattei, M.: Robust fault detection and isolation for proprioceptive sensors of robot manipulators. Mechatronics **20(1)**, 162–170 (2010)
20. Pezzato, C., Ferrari, R., Corbato, C.H.: A novel adaptive controller for robot manipulators based on active inference. IEEE Robotics and Automation Letters (2020)
21. Pio-Lopez, L., Nizard, A., Friston, K., Pezzulo, G.: Active inference and robot control: a case study. Journal of The Royal Society Interface **13(122)** (2016)
22. Sancaktar, C., Lanillos, P.: End-to-end pixel-based deep active inference for body perception and action. arXiv preprint arXiv:2001.05847 (2019)
23. Van, M., Wu, D., Ge, S., Ren, H.: Fault diagnosis in image-based visual servoing with eye-in-hand configurations using Kalman filter. IEEE Transactions Industrial Electronics **12(6)**, 1998–2007 (2016)
24. Vanderbroeck, M., Baioumy, M., van der Lans, D., de Rooij, R., van der Werf, T.: Active inference for robot control: A factor graph approach. Student Undergraduate Research E-journal! **5**, 1–5 (2019)
25. Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems. Annual Reviews in Control **32**(2), 229 – 252 (2008)

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

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

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

Title of Paper	Active Inference for Fault Tolerant Control of Robot Manipulators with Sensory Faults
Publication Status	Published
Publication Details	Pezzato, Corrado, 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*, pp. 20-27. Springer International Publishing, 2020.

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	The idea for the paper was jointly developed by me and Corrado Pezzato. Corrado Pezzato led the work on the presented method and experimental validation. I led the work on analyzing the results and identifying the limitations of this approach. I co-led the writing of this manuscript with Corrado Pezzato, in collaboration with Nick Hawes, Carlos Hernandez and Martijn Wisse.		
Signature 		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

B. ***Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference***

This paper presents a novel fault-tolerant control scheme for robotic systems using active inference. We use an enhanced active inference controller, named the unbiased Active Inference Control (u-AIC), which offers unbiased state estimation and simplifies the definition of probabilistically robust thresholds for fault detection and isolation (FDI). This approach efficiently handles sensory faults without the need for additional recovery controllers. Demonstrated through simulations on a 2-DOF manipulator, our method effectively simplifies fault detection processes, while significantly reducing the chances of false positives.

Now we discuss the potential utility of utilizing probabilistic inference in this paper

- **Novel Capabilities (✓):** This paper presents a robust approach for fault-tolerant control. Similar to previous work, it is computationally efficient and simplifies fault-detection and isolation (FDI). Additionally, this approach is robust and significantly reduces the chances of false positives, making it viable for real-world applications.
- **Coherence (✓):** Coherence in this work makes many operations straightforward to achieve. For instance, Fault-isolation is greatly simplified by separating the free-energy function for every sensor (see equations 25 and 26). This reduces system complexity and design.
- **Joint Optimization (✓) :** Similar to previous work, joint optimization makes many computations redundant, which results in substantial computation gains.
- **Automated Solvers (×):** The thresholds of fault-detection were implemented independently, and thus automated solvers were not used in this case. This was a design choice for simplicity rather than a fundamental limitation. However, a standard solver was used for the u-AIC.
- **Connection to Literature (×) :** This paper doesn't show any new connections to different fields.

The paper starts on the following page.

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

Mohamed Baioumy[1*], Corrado Pezzato[2*], Riccardo Ferrari[2], Carlos Hernández Corbato[2] and Nick Hawes[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.**

***Index Terms*— Fault-tolerant control, fault recovery, active inference, free-energy principle, robotics, robot manipulator.**

I. Introduction

Fault tolerant (FT) control is of vital importance for many applications such robots working in production lines [1]. In order to guarantee high standards of reliability and security, the area of FT control of robot manipulators increasingly gathered importance in recent years [2]. Many approaches have been developed for faults in actuators and sensors [3]–[7]. Model based fault tolerant techniques are amongst the most advanced approaches to tackle the problem of fault detection and isolation (FDI) for dynamical systems [8]. 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 [9].

Regarding FT control for robot manipulators, [6] performs 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 lack of observability in case of sensory faults. In [7], three different second-order sliding mode observers are used 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 [10]–[13]. The work in [14] showed fault-tolerant steering control of an agricultural vehicle for bailing, detecting sensory faults and artefacts in images through statistical learning. The sensors configuration for control was then decided at run-time.

Generally speaking, model-based fault tolerant control schemes rely on additional 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 in [15], [16]. Active inference relies on approximate Bayesian inference [17], [18] for prediction errors 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, see [15], [19] respectively. The authors in [20] initially presented a fault tolerant control scheme with an AIC which showed great promise. However, this presented a few fundamental limitations which will be thoroughly discussed in Sec. II-B. Most importantly, FDI in [20] can be affected by false-positives when 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 organised as follow: Sec. II presents the background on active inference for fault-tolerant control highlighting the limitations of past work. Sec. III presents a new formulation of the AIC with unbiased state estimation, while Sec. IV explains how fault tolerant control is achieved with this new formulation. Results for a simulated 2-DOF robot arm are presented in Sec. V while Sec. VI reports a summary and future challenges.

*These authors contributed equally to this work.

[1] Authors are with the Oxford Robotics Institute, University of Oxford. For correspondence {mohamed, nickh}@robots.ox.ac.uk

[2] Authors are with Delft University of Technology. For correspondace {c.pezzato, c.h.corbato, r.ferrari}@tudelft.nl

This work was supported by Ahold Delhaize

II. PRELIMINARIES

This section presents the background knowledge on active inference for robot control [15], [20] required to understand and justify the need for an unbiased AIC.

A. Active inference controller (standard AIC)

The AIC in [15], [20] is defined for torque control in joint space of a generic robot manipulator (Fig. 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 on the basis of the so-called *free-energy* and the *generalised motions* $\tilde{\boldsymbol{\mu}}$ [20].

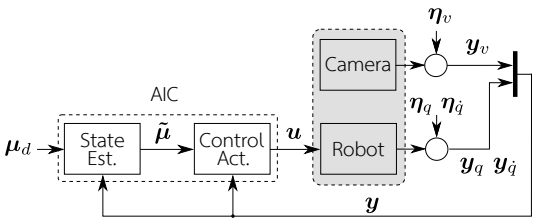


Fig. 1. General AIC control scheme for joint space torque control.

The free-energy principle relies on Bayesian inference [21] 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 $Q(\boldsymbol{x})$ [22]. The most probable state is computed by minimizing the Kullback-Leibler divergence (D_{KL}) between $p(\boldsymbol{x}|\boldsymbol{y})$ and $Q(\boldsymbol{x})$ [18]:

$$\begin{aligned} D_{KL}(Q(\boldsymbol{x})||p(\boldsymbol{x}|\boldsymbol{y})) &= \int Q(\boldsymbol{x}) \ln \frac{Q(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y})} d\boldsymbol{x} \\ &= F + \ln p(\boldsymbol{y}) \end{aligned} \tag{1}$$

where F represents the *free-energy*. By minimizing 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:

$$p(\boldsymbol{x}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) = \underbrace{p(\boldsymbol{y}_v|\boldsymbol{x}) p(\boldsymbol{y}_q|\boldsymbol{x}) p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{2}$$

If $Q(\boldsymbol{x})$ is a Gaussian with mean $\boldsymbol{\mu}$ and the Laplace approximation is utilized [23], then F reduces to

$$F = -\ln p(\boldsymbol{\mu}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) + C\,, \tag{3}$$

where C 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* [18], [24]. 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 according to [18] as $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{\mu}) + \boldsymbol{z}$, 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 [18]:

$$\frac{d\boldsymbol{\mu}}{dt} = \boldsymbol{\mu}' = \boldsymbol{f}(\boldsymbol{\mu}) + \boldsymbol{w} \tag{4}$$

where $\boldsymbol{f}(\boldsymbol{\mu}) : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a generative function dependant 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 [16], 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.

$$\boldsymbol{f}(\boldsymbol{\mu}) = (\boldsymbol{\mu}_d - \boldsymbol{\mu})\tau^{-1} \tag{5}$$

From equation (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 [16], 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 [16], [25]. 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 (7). If all distributions in equation (2) are assumed Gaussian, the expression for free-energy simplifies to (see [15] for complete derivations):

$$F = \frac{1}{2} \sum_i \left(\boldsymbol{\varepsilon}_i^\top P_i \boldsymbol{\varepsilon}_i - \ln |P_i| \right) + C, \tag{6}$$

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 [15], [26]. 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 $\boldsymbol{\varepsilon}_{\mu'} = (\boldsymbol{\mu}'' - {}^{\partial \boldsymbol{f}(\mu)}/_{\partial \boldsymbol{\mu} \mu'})$. 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 (6), refer to [15], [16], [18].

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

$$\dot{\tilde{\boldsymbol{\mu}}} = D\tilde{\boldsymbol{\mu}} - \kappa_\mu \frac{\partial F}{\partial \tilde{\boldsymbol{\mu}}}, \tag{7}$$

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 [17]. The control actions, are also computed through gradient descent; however, the expression

for F does not include any actions explicitly. The chain rule is then used, assuming an implicit dependency between actions and sensory input.

$$\dot{\boldsymbol{u}} = -\kappa_u \frac{\partial F}{\partial \boldsymbol{u}} = -\kappa_u \frac{\partial F}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}}, \tag{8}$$

where κ_u is the gradient descent's step size. The term $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}}$ is often approximated as linear, similarly to [15], [26].

B. Limitations of fault-tolerant control with standard AIC

The AIC was used in [20] for fault tolerant control. In this subsection we aim at highlighting the main limitations of the AIC, which are addressed in Sec. III through the proposed unbiased active inference scheme. The main idea in [20] 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 to sensory readings. For a faulty sensor, the relative precision was decreased to zero to perform fault recovery. Specifically, the residuals in [20] are generated by considering the quadratic form of the SPE $\boldsymbol{\varepsilon}_s^\top P_{y_s} \boldsymbol{\varepsilon}_s$, 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 [20] because of the over-conservative threshold used for fault detection, but it becomes apparent when the SPE are closely analysed. Fig. 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 $\boldsymbol{\mu}_{d_1}$, $\boldsymbol{\mu}_{d_2}$, $\boldsymbol{\mu}_{d_3}$.

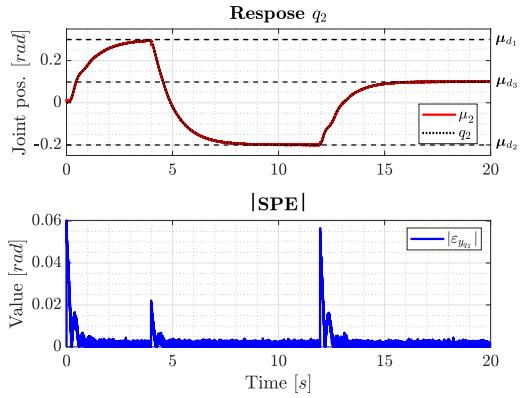


Fig. 2. Effect of abrupt change in goal on the SPE in standard AIC. The SPE increases sharply without the presence of any sensory faults.

Using a tighter threshold to detect faults of small entity by using the approach proposed in [20] 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 SPE are independent from 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 [20].

III. Unbiased active inference

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

A. 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:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \underbrace{p(\boldsymbol{y}_v|\boldsymbol{x}) p(\boldsymbol{y}_q|\boldsymbol{x}) p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{9}$$

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:

$$F = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}_v, \boldsymbol{y}_q, \boldsymbol{y}_{\dot{q}}) + C \tag{10}$$

This expression can be factorized as in equation (9). Assuming Gaussian model F can be expanded to:

$$\begin{aligned} F = \frac{1}{2}(&\boldsymbol{\varepsilon}_{y_q}^\top P_{y_q} \boldsymbol{\varepsilon}_{y_q} + \boldsymbol{\varepsilon}_{y_{\dot{q}}}^\top P_{y_{\dot{q}}} \boldsymbol{\varepsilon}_{y_{\dot{q}}} + \boldsymbol{\varepsilon}_{y_v}^\top P_{y_v} \boldsymbol{\varepsilon}_{y_v} \\ &+ \boldsymbol{\varepsilon}_x^\top P_x \boldsymbol{\varepsilon}_x + \boldsymbol{\varepsilon}_u^\top P_u \boldsymbol{\varepsilon}_u - \ln |P_u P_{y_q} P_{y_{\dot{q}}} P_{y_v} P_x|) + C, \end{aligned} \tag{11}$$

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:

$$\hat{\boldsymbol{x}}_{k+1} = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \boldsymbol{\mu}_{x,k} \tag{12}$$

where I represents an 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 actions update, first-order Euler integration is used to obtain discrete time equations.

B. 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 [19]. 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 by 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:

$$k(\boldsymbol{y}_{q_i}, \boldsymbol{y}_{q_j}) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{y}_{q_i} - \boldsymbol{y}_{q_j})^\top \Theta (\boldsymbol{y}_{q_i} - \boldsymbol{y}_{q_j})\right) + \sigma_n^2 d_{ij} \tag{13}$$

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, see [27]. The predictions are then [19]:

$$\begin{aligned} \boldsymbol{g}_v(\boldsymbol{y}_{\boldsymbol{q}*}) &= \begin{bmatrix} k(\boldsymbol{y}_{\boldsymbol{q}*}, \bar{\boldsymbol{y}}_q) K^{-1} \bar{\boldsymbol{y}}_{v_x} \\ k(\boldsymbol{y}_{\boldsymbol{q}*}, \bar{\boldsymbol{y}}_q) K^{-1} \bar{\boldsymbol{y}}_{v_z} \end{bmatrix} \\ \boldsymbol{g}_v(\boldsymbol{y}_{\boldsymbol{q}*})' &= \begin{bmatrix} -\Theta^{-1}(\boldsymbol{y}_{\boldsymbol{q}*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{\boldsymbol{q}*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_x] \\ -\Theta^{-1}(\boldsymbol{y}_{\boldsymbol{q}*} - \bar{\boldsymbol{y}}_q)^\top [k(\boldsymbol{y}_{\boldsymbol{q}*}, \bar{\boldsymbol{y}}_q)^\top \cdot \boldsymbol{\alpha}_z] \end{bmatrix} \end{aligned} \tag{14}$$

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

C. Estimation and control using the u-AIC

Similar to the AIC, performing both state-estimation and control can be achieved using gradient descent on 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. (8)) or to consider generalized motion (7):

$$\dot{\boldsymbol{\mu}}_u = -\kappa_u \frac{\partial F}{\partial \boldsymbol{\mu_u}}, \quad \dot{\boldsymbol{\mu}}_x = -\kappa_\mu \frac{\partial F}{\partial \boldsymbol{\mu_x}}, \tag{15}$$

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

$$\frac{\partial F}{\partial \boldsymbol{\mu}_u} = P_u(\boldsymbol{\mu}_u - f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)) \tag{16}$$

Setting this to zero results in the control action being equal to $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$. If this function is chosen similar 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. (9). The only term depending on the control action now is $p(\boldsymbol{u}|\boldsymbol{x})$; however, a prior $p(\boldsymbol{u})$ can be also added. In that case the generative model would be:

$$\frac{1}{\alpha} p(\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{x}) p(\boldsymbol{y}_v|\boldsymbol{x}) p(\boldsymbol{y}_q|\boldsymbol{x}) p(\boldsymbol{y}_{\dot{q}}|\boldsymbol{x}) p(\boldsymbol{x}) \tag{17}$$

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 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 in [28].

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 sensor, exactly as in a Kalman filter [29].

D. Key differences between AIC and u-AIC

The AIC considers the relationship between states and observations without explicitly modelling the control actions. The prior over state $p(\boldsymbol{x})$ in equation (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 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. (5)).

In the u-AIC, actions are explicitly modelled 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 a prediction step will also improve the overall state-estimation accuracy. Secondly, explicit actions allow us to directly perform gradient descent on 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. (8)).

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

IV. 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 (12),

and expanding the second term in (15) where F is computed as in (11), the state estimation law can be rewritten as:

$$\boldsymbol{\mu}_{k+1} = \gamma(\boldsymbol{\mu}_k, \boldsymbol{u}_k) + \Lambda(\boldsymbol{g}(\boldsymbol{\mu}_k) - \boldsymbol{y}_k) \tag{18}$$

Equation (18) represents the dynamics of an estimator where stability results from the value that we obtain for diagonal matrix Λ. In [29] it has been shown how the free-energy principle can be used to derive stable state observers, for a linear case with coloured noise. It is important to note that γ and Λ are known expressions resulting from the partial derivatives of 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:

$$\begin{cases} \boldsymbol{x}_{k+1} = \gamma(\boldsymbol{x}_k, \boldsymbol{u}_k) + \phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho}_k) \\ \boldsymbol{y}_k = \boldsymbol{x}_k + \boldsymbol{z}_k \end{cases} \tag{19}$$

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 easily a model-based fault diagnosis approach 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 $\phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{0}) = 0$.

A. Residual generation

Following [12], [30] we rely on two assumptions:

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 $(\mathcal{Z}, \mathscr{B}, \mathbf{P}_{\mathcal{Z}})$, where $\mathcal{Z} \subseteq \mathbb{R}^n$. $\mathscr{B}(\cdot)$ indicates a Borel σ-algebra, and $\mathbf{P}_{\mathcal{Z}}$ is the probability measures defined over $\mathcal{Z}$. Furthermore, it is not correlated and are independent from $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ 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 (19), (18) and following [12], [30], the residuals dynamics will be given by:

$$\boldsymbol{r}_{k+1} = \Lambda \boldsymbol{r}_k + \boldsymbol{\delta}_k + \phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho}_k) \tag{20}$$

$$\boldsymbol{\delta}_k = \gamma(\boldsymbol{y}_k - \boldsymbol{z}_k, \boldsymbol{u}_k) - \gamma(\boldsymbol{\mu}_k, \boldsymbol{u}_k) + \boldsymbol{z}_{k+1} \tag{21}$$

The stochastic process $\boldsymbol{\delta}_k$, is the random total uncertainty which 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 $\boldsymbol{\delta}_k$ [12].

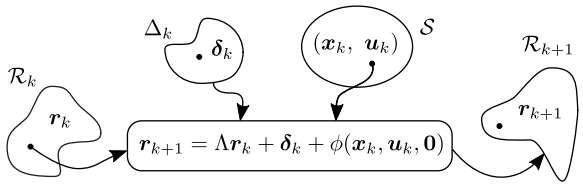


Fig. 3. Healthy residual set at time $k+1$ as image obtained from the output of (20).

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 (20), where the total uncertainty $\boldsymbol{\delta}_k$ varies over its domain Δ_k (Fig. 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 the healthy and faulty behaviours. In particular, γ is obtained via algebraic manipulations from 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.

B. Threshold for fault detection

As in [30], we consider a probabilistically robust detection logic of the form:

$$d_M(\boldsymbol{r}_{k+1}) \leq \frac{n}{\alpha} \triangleq \overline{d_M} \tag{22}$$

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:

$$d_M(\boldsymbol{r}) = \sqrt{(\boldsymbol{r} - \bar{\boldsymbol{r}})^\top C_r^{-1} (\boldsymbol{r} - \bar{\boldsymbol{r}})} \tag{23}$$

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 [31]:

$$Pr[d_M(\boldsymbol{r}_k) \geq \frac{n}{\alpha}] \leq \alpha, \qquad \forall \alpha \in [0,\ 1] \tag{24}$$

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 check 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.

C. Fault isolation

The detection policy in (22) results effective to label the system as healthy or faulty. However, it is not possible to use the same SPE generated through equation (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 (11). We define two additional free-energies and state estimation processes which rely on different sensory sources (i.e. either proprioceptive or visual).

$$F_p = \frac{1}{2}(\boldsymbol{\varepsilon}_{\boldsymbol{y_q}}^\top P_{y_q} \boldsymbol{\varepsilon}_{\boldsymbol{y_q}} + \boldsymbol{\varepsilon}_{\boldsymbol{y_{\dot{q}}}}^\top P_{y_{\dot{q}}} \boldsymbol{\varepsilon}_{\boldsymbol{y_{\dot{q}}}} + \boldsymbol{\varepsilon}_{\boldsymbol{x}}^\top P_x \boldsymbol{\varepsilon}_{\boldsymbol{x}} - \ln|P_{y_q} P_{y_{\dot{q}}} P_x|), \tag{25}$$

$$F_v = \frac{1}{2}(\boldsymbol{\varepsilon}_{\boldsymbol{y_v}}^\top P_{y_v} \boldsymbol{\varepsilon}_{\boldsymbol{y_v}} + \boldsymbol{\varepsilon}_{\boldsymbol{x}}^\top P_x \boldsymbol{\varepsilon}_{\boldsymbol{x}} - \ln|P_{y_v} P_x|) \tag{26}$$

Doing so, we can isolate encoder and camera faults since state estimation is now done using two independent measurement sources. F_p and F_v are only used for fault isolation and not control. The thresholds for the SPE from (25) and (26) are then computed as in Sec. IV-B.

D. Fault recovery

Fault recovery can be performed as in [20]. 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) to 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:

$$P_{fs} = \mathbf{0} \tag{27}$$

Interestingly, the active inference framework allows also for hyper-parameters (precision) learning [16], [32]. The bias in the standard AIC hindered hyper-parameters 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.

V. 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 [33]:

$$\boldsymbol{u} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + D\dot{\boldsymbol{q}} + G(\boldsymbol{q}) \tag{28}$$

where $\boldsymbol{q} = [q_1,\ q_2]^\top$, $\boldsymbol{u}(t) = [u_1,\ u_2]^\top$, 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, 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 $\boldsymbol{q} = [-\pi/2,\ 0]\ (rad)$ to $\boldsymbol{\mu}_{d1} = [-0.2,\ 0.5]\ (rad)$, and then $\boldsymbol{\mu}_{d2} = [-0.6,\ 0.2]\ (rad)$. A fault occurs during the trajectory from $\boldsymbol{\mu}_{d1}$ to $\boldsymbol{\mu}_{d2}$, 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 $\boldsymbol{y}_q(k) = [q_1(k_f),\ q_2(k)]^\top$ 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 Fig. 4 and Fig. 5:

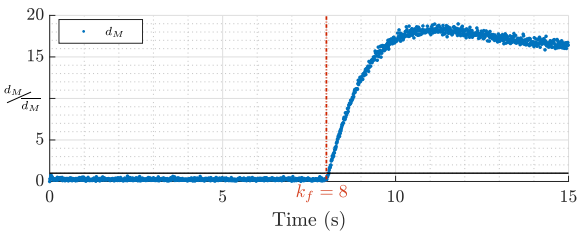


Fig. 4. Normalised d_M in case of encoder fault. A detection occur when the normalized value exceeds 1.

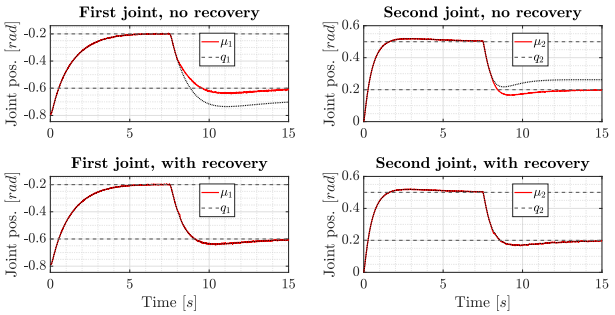


Fig. 5. System's response in case of encoder fault without recovery (fixed precision matrices) and with recovery (setting $P_{y_q} = \mathbf{0}$ after detection). The fault occurs after 8s.

As can be seen from Fig. 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 minimise 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. $\boldsymbol{y}_v(k) = \boldsymbol{y}_v(k) + 0.04$ 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 Fig. 6. The system's response with and without recovery is similar to Fig. 5.

The simulation results will be extended to real experiments, leveraging the scalability and adaptability of the active inference controller [15]. The steady state errors (e_{ss}) and

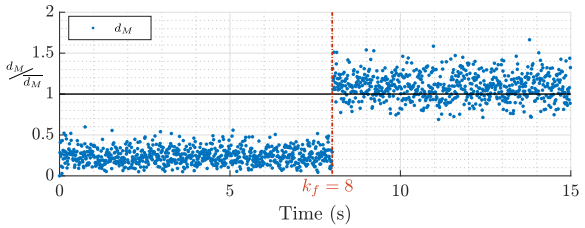


Fig. 6. Normalised d_M in case of camera fault.

the $RMSE$ between beliefs and actual joint positions for the simulations are reported in Table I.

TABLE I

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	$3.5e^{-4}$	$2.1e^{-3}$

VI. 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 for computing partial derivatives of the free-energy. Next to that, 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.

REFERENCES

[1] J. Krüger, T. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals - Manufacturing Technology*, 2009.

[2] M. Visinsky, J. Cavallaro, and I. Walker, "Robotic fault detection and fault tolerance: A survey," *Reliability Engineering and System Safety*, vol. 46, pp. 139–158, 1994.

[3] W. Dixon, I. Walker, D. Dawson, and J. Hartranft, "Fault detection for robot manipulators with parametric uncertainty: A prediction-error-based approach," *IEEE Trans. on Robotics and Automation*, vol. 16(6), pp. 689–699, 2000.

[4] J. Shin and J. Lee, "Fault detection and robust fault recovery control for robot manipulators with actuator failures," in *Procs. of the IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 861–866.

[5] G. Paviglianiti, F. Pierri, F. Caccavale, and M. Mattei, "Robust fault detection and isolation for proprioceptive sensors of robot manipulators," *Mechatronics*, vol. 20(1), pp. 162–170, 2010.

[6] M. Van, D. Wu, S. Ge, and H. Ren, "Fault diagnosis in image-based visual servoing with eye-in-hand configurations using Kalman filter," *IEEE Trans. Industrial Electronics*, vol. 12(6), pp. 1998–2007, 2016.

[7] M. Capisani, A. Ferrara, and P. Pisu, "Sliding mode observers for vision-based fault detection, isolation and identification in robot manipulators," in *American Control Conference*, 2010.

[8] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, LLC, 1999.

[9] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Transaction on Automatic Control*, 1997.

[10] S. Fang, M. Blanke, and B. J. Leira, "Mooring system diagnosis and structural reliability control for position moored vessels," *Control engineering Practice*, vol. 36, pp. 12–26, 2015.

[11] R. Ferrari, H. Dibowski, and S. Baldi, "A message passing algorithm for automatic synthesis of probabilistic fault detectors from building automation ontologies," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4184–4190, 2017.

[12] V. Rostampour, R. Ferrari, and T. Keviczky, "A set based probabilistic approach to threshold design for optimal fault detection," *Procs. of the American Control Conference*, pp. 5422–5429, 2017.

[13] V. Rostampour, R. M. Ferrari, A. M. Teixeira, and T. Keviczky, "Privatized distributed anomaly detection for large-scale nonlinear uncertain systems," *IEEE Transactions on Automatic Control*, 2020.

[14] M. R. Blas and M. Blanke, "Stereo vision with texture learning for fault-tolerant automatic baling," *Computers and electronics in agriculture*, vol. 75, no. 1, pp. 159–168, 2011.

[15] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, 2020.

[16] M. Baioumy, P. Duckworth, B. Lacerda, and N. Hawes, "Active inference for integrated state-estimation, control, and learning," arXiv preprint arXiv:2005.05894, 2020.

[17] K. J. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11(2), pp. 27–138, 2010.

[18] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.

[19] P. Lanillos and G. Cheng, "Adaptive robot body learning and estimation through predictive coding," in *IROS*, 2018.

[20] C. Pezzato, M. Baioumy, C. H. Corbato, N. Hawes, M. Wisse, and R. Ferrari, "Active inference for fault tolerant control of robot manipulators with sensory faults," in *1st Int. Workshop on Active Inference, ECML PKDD*, ser. Communications in Computer and Information Science, Springer, Ed., vol. 1326, 2020.

[21] D. Lindley, "Bayesian statistics, a review," *SIAM*, vol. 2, 1972.

[22] C. W. Fox and S. J. Roberts, "A tutorial on variational bayesian inference," *Artificial intelligence review*, vol. 38, no. 2, pp. 85–95, 2012.

[23] K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the Laplace approximation," *Neuroimage*, vol. 34(1), pp. 220–234, 2007.

[24] K. J. Friston, J. Mattout, and J. Kilner, "Action understanding and active inference," *Biological cybernetics*, vol. 104(1-2), pp. 137–160, 2011.

[25] M. Baltieri and C. L. Buckley, "A probabilistic interpretation of PID controllers using active inference," in *Int. Conference on Simulation of Adaptive Behavior*. Springer, 2018, pp. 15–26.

[26] G. Oliver, P. Lanillos, and G. Cheng, "Active inference body perception and action for humanoid robots," arXiv preprint arXiv:1906.03022v2, 2019.

[27] C. Rasmussen and C. Williams, Eds., *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[28] M. Baioumy, M. Mattamala, and N. Hawes, "Variational inference for predictive and reactive controllers," in *ICRA 2020 Workshop on New advances in Brain-inspired Perception, Interaction and Learning*, 2020.

[29] A. Meera and M. Wisse, "Free energy principle based state and input observer design for linear systems with colored noise," in *2020 American Control Conference (ACC)*, 2020, pp. 5052–5058.

[30] V. Rostampour, R. Ferrari, A. M. Teixeira, and T. Keviczky, "Differentially-Private Distributed Fault Diagnosis for Large-Scale Nonlinear Uncertain Systems," *IFAC-PapersOnLine*, vol. 51, pp. 975–982, 2018.

[31] X. Chen, "A new generalization of Chebyshev inequality for random vectors," *arXiv preprint arXiv:0707.0805*, 2007.

[32] R. Bogacz, "A tutorial on the free-energy framework for modelling perception and learning," *Journal of mathematical psychology*, 2015.

[33] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

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

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

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

Title of Paper	Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference
Publication Status	Published
Publication Details	Mohamed Baioumy, Corrado Pezzato, Riccardo Ferrari, Carlos H. Corbato and Nick Hawes, "Fault-tolerant Control of Robot Manipulators with Sensory Faults using Unbiased Active Inference", European Control Conference (ECC), 2021

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	The idea for the paper was jointly developed by me and Corrado Pezzato. Corrado and Riccardo performed the literature review on Fault-tolerant control. I led the development of the novel strategy presented. I led the implementation of the novel strategy, while Corrado led the implementation of existing benchmarks. I co-led the writing of the paper with Corrado, in collaboration with Nick Hawes, Carlos Hernandez and Riccardo Ferrari.		
Signature 		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

C. *Beta residuals: improving fault-tolerant control for sensory faults via Bayesian inference and precision learning*

This paper investigates a Bayesian approach to fault-tolerant control (FTC) in robotics. This approach aims to unify fault detection, isolation, and accommodation into a single Bayesian inference framework. We introduce a precision-learning based Bayesian FTC method and a novel 'beta residual' for fault detection. This approach streamlines the FTC process by integrating implicit fault recovery and providing interpretable insights into sensor faults, without requiring explicit fault detection steps. Simulation results validate the effectiveness of this method. The main contributions include: 1. A Bayesian controller for stochastic FTC capable of tracking variations and faults through precision learning. 2. The introduction of 'beta residual', an interpretable metric within the Bayesian inference process, indicating sensor faultiness.

Next, we discuss the potential utility of utilizing probabilistic inference in this paper.

- **Novel Capabilities (✓):** This paper presents a novel approach for fault-tolerant control outperforming existing methods. The approach is accurate, computationally efficient, and simple. We additionally, introduce a 'beta residuals': a novel residual signal.
- **Coherence (✓):** A key in this paper is coherence across the whole stack. Control, state-estimation, fault-detection and fault-recovery are all modelled a single inference problem. This greatly simplifies the system design and aids in us finding new insights such as the beta residual.
- **Joint Optimization (✓) :** This approach is the most accurate compared all benchmarks, yet it is extremely computationally efficient. This is largely due to joint optimization across the stack.
- **Automated Solvers (✓):** A standard factor graph library (ForneyLab [33]) was used to solve all problems in this paper. No modifications or additional considerations were needed as the solver manages all that complexity.
- **Connection to Literature (✓) :** This paper connects ideas in Bayesian Inference to fault-tolerant control. The beta residual is measure of uncertainty of a gamma distribution and is related to adaptive residual signals in fault-detection.

The paper starts on the following page.

Beta Residuals: Improving Fault-Tolerant Control for Sensory Faults via Bayesian Inference and Precision Learning

Mohamed Baioumy [*] **William Hartemink** [**]
Riccardo M.G. Ferrari [***] **Nick Hawes** [*]

[*] *Oxford Robotics Institute, University of Oxford, United Kingdom. For correspondence: mohamed@robots.ox.ac.uk.*
[**] *Amazon Web Services, Texas, United States.*
[***] *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands. (e-mail: r.ferrari@tudelft.nl)*

Abstract: Model-based fault-tolerant control (FTC) often consists of two distinct steps: fault detection & isolation (FDI), and fault accommodation. In this work we investigate posing fault-tolerant control as a single Bayesian inference problem. Previous work showed that precision learning allows for stochastic FTC without an explicit fault detection step. While this leads to implicit fault recovery, information on sensor faults is not provided, which may be essential for triggering other impact-mitigation actions. In this paper, we introduce a precision-learning based Bayesian FTC approach and a novel *beta residual* for fault detection. Simulation results are presented, supporting the use of beta residual against competing approaches.

1. INTRODUCTION

The emergence of autonomous robotic systems calls for the introduction of fault-tolerant control architectures that can guarantee safe operation even under faulty conditions. *Fault-Tolerant Control* (FTC) techniques can limit the impact of faults at process, sensor or actuator level (Blanke et al., 2000; Chen and Patton, 1999). FTC approaches typically consists of two steps: fault detection, isolation and identification (FDI); and fault accommodation.

Model-based approaches to FTC proved to be very powerful (Gao et al., 2015), but requires high amounts of knowledge and data for modelling the system and designing FDI residuals and thresholds. This holds especially for cases modelled as stochastic systems, where probabilistic rather than deterministic solutions are preferred.

A notable contribution to solving this problem has been provided by the prolific literature on Bayesian approaches to FDI. Cai et al. (2017) provides an overview of early works which used static Bayesian Networks to model the causal relation between faults and symptoms, and obtain a diagnosis via inference. Later works introduced Dynamic Bayesian Networks (DBN) as a way to encode transient and time varying behaviours, and general spatial and time relations. For instance Lerner et al. (2000) and Verma et al. (2004) use, respectively, a Kalman and a Particle Filter to track possible fault hypotheses, while Codetta-Raiteri and Portinale (2015) illustrate the use of Probabilistic Graphic Models (PGM) and DBNs for spacecraft FDI. In Sun et al. (2020) a Bayesian Recurrent Neural Network is used as a probabilistic model, whose posterior is approximated via a variational approach.

Still, several challenges remain at the forefront of research on this topic. On one side, an integrated Bayesian approach for both FDI and fault recovery is sought. On another, the computational complexity of inference over DBN can be unsuited to embedded implementations on board autonomous robots, when complex models are employed. Finally, Bayesian approaches need adaptation capabilities to cope with time varying systems, for instance due to ageing or other non-fault phenomena.

In order to address the first two issues, several works in the field of robotics and control have recently taken inspiration from the *active inference* framework. In neuroscience, it is regarded as a general theory for perception and action (Friston, 2010) and can be used to understand decision making of biological agents and to build artificial agents. Active inference methods for control have shown promising results in deterministic FTC (Pezzato et al., 2020; Baioumy et al., 2021c), stochastic FTC (Baioumy et al., 2021b), adaptive control for robot manipulators (Buckley et al., 2017; Pezzato et al., 2020; Baioumy et al., 2021a) and state-estimation (Lanillos and Cheng, 2018; Friston et al., 2010).

In this work we address the third challenge as well, by presenting a Bayesian controller for stochastic fault-tolerant control that can track naturally occurring variations and faults via *precision learning*. . In particular, we model the precision (inverse covariance) of each sensor as a random variable and compute on-line its posterior. The *expected precision* acts as a measure of the probability of a sensor being faulty, and its inverse is used to weight a given sensor measurement in the control law. This approach does not require a-priori information about fault characteristics, or an explicitly-defined fault detection criteria and recovery

mechanisms. Still, operators may need an interpretable estimate of the location and probability of a fault. To address this, we introduce a so-called *beta residual* based on the estimation of hyper-parameters of the precision distribution density. A corresponding residual evaluation based on the logistic function is then proposed.

The main contributions of this paper can then be summarised as

(1) a set of controllers performing *sensor* FTC as Bayesian inference by means of precision learning;
(2) a novel approach for extracting an *interpretable* quantity, the beta-residual, representing the controller's Bayesian belief of sensor faultiness.

The rest of the paper is organized as following: Section 2 introduces two types of Bayesian control methods which are used in Section 3 to implement FTC via precision learning. The interpretable beta residual is defined in Section 4, while simulation results and conclusions are presented, respectively, in Section 5 and 6.

2. CONTROL MODELS

In this section, we recall results from prior work to motivate the work proposed in this paper. We first formalize the system dynamics in a problem statement; then, we present two controllers based on Bayesian inference: the unbiased active inference controller (u-AIC) of (Baioumy et al., 2021c,b) and a general Bayesian Controller (BC).

2.1 Problem statement

Let us consider a discrete-time nonlinear system as

$$\begin{cases} \boldsymbol{x}_{t+1} & = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{q} \\ \boldsymbol{y}_t & = g(\boldsymbol{x}_t) + \boldsymbol{\eta} \end{cases}, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$, $\boldsymbol{u}_t \in \mathbb{R}^{N_u}$ and $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$ are, respectively, the system's state, input and output vector. The variables $\boldsymbol{q} \in \mathbb{R}^{N_x}$ and $\boldsymbol{\eta} \in \mathbb{R}^{N_y}$ are zero-mean, Gaussian process and measurement noises, instead. The controller aims to steer the system to a goal state $\boldsymbol{x}_{\text{goal}}$ by applying a control action $\boldsymbol{u}_t$. At any time, unknown to the controller, sensor faults can occur. This includes *hard faults* (complete failure) and *soft faults* (e.g. arbitrary sensor drifts or constant offsets).

2.2 The Unbiased Active Inference Controller

The (u-AIC) uses a generative probabilistic model with the current state $\boldsymbol{x}$, control action $\boldsymbol{u}$ and observations $\boldsymbol{y}$. The system can have multiple observation modalities. For instance, a robot can have joint encoders for position and velocity in addition to a camera. As a running example throughout the paper, we derive the equations for a system with two joint encoders $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}$ and a visual sensor $\boldsymbol{y}^{(v)}$. In this case the joint distribution $p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)})$ is assumed to be factorized as:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \underbrace{p(\boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{2}$$

Given the sensor data, we then aim to find the posterior over states and actions $p(\boldsymbol{x}, \boldsymbol{u}|\boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)})$. As is common in variational Bayesian inference, we approximate the posterior with a variational distribution $q(\boldsymbol{x}, \boldsymbol{u})$ and utilize the mean-field assumption ($q(\boldsymbol{x}, \boldsymbol{u}) = q(\boldsymbol{x})q(\boldsymbol{u})$) and the Laplace approximation Fox and Roberts (2012). 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 expression for *variational free-energy*, a quantity which is then minimized to generate control actions:

$$F = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}) + C. \tag{3}$$

Detailed derivations of the above are available in Baioumy et al. (2021c), while more general information about variational inference is available in Murphy (2012). It then follows, assuming Gaussian distributions, that F can be factorized as in eq. (2) and then expanded to

$$\begin{aligned} F = \frac{1}{2}(&\boldsymbol{\varepsilon}_{y^{(1)}}^\top P_{y^{(1)}} \boldsymbol{\varepsilon}_{y^{(1)}} + \boldsymbol{\varepsilon}_{y^{(2)}}^\top P_{y^{(2)}} \boldsymbol{\varepsilon}_{y^{(2)}} + \boldsymbol{\varepsilon}_{y^{(v)}}^\top P_{y_v} \boldsymbol{\varepsilon}_{y^{(v)}} \\ &+ \boldsymbol{\varepsilon}_x^\top P_x \boldsymbol{\varepsilon}_x + \boldsymbol{\varepsilon}_u^\top P_u \boldsymbol{\varepsilon}_u - \ln |P_u P_{y^{(1)}} P_{y^{(2)}} P_{y^{(v)}} P_x|) + C, \end{aligned} \tag{4}$$

where $\boldsymbol{\varepsilon}_{y^{(1)}}$, $\boldsymbol{\varepsilon}_{y^{(2)}}$, $\boldsymbol{\varepsilon}_{y^{(v)}}$ are the sensor 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 on the control action.

The prediction error on the state prior $\boldsymbol{\varepsilon}_x = (\boldsymbol{\mu}_x - \hat{\boldsymbol{x}})$ is computed from $\hat{\boldsymbol{x}}$, the prediction of the state, which is a deterministic value. The prediction can be computed via the prediction step of a Kalman filter: an advantage of the (u-AIC) is that an accurate model is not required. In this paper a simple random walk is assumed.

Finally, the information about the target/goal state is encoded in the control bias 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 proportional (P) controller: $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)) = P(\boldsymbol{\mu}_d - \boldsymbol{\mu}_x)$.

2.3 Bayesian Controller

We now consider a Bayesian controller as in Baioumy et al. (2020), whose generative model is defined as

$$p(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{y}_t, \boldsymbol{u}_t) \propto \underbrace{p(\boldsymbol{x}_t)}_{\text{Prediction prior}} \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)}_{\text{Transition model}} \underbrace{p(\boldsymbol{y}_t|\boldsymbol{x}_t)}_{\text{Observation model}} \underbrace{p(\boldsymbol{x}_{t+1})}_{\text{Goal prior}}. \tag{5}$$

The prediction prior $p(\boldsymbol{x}_t)$ is defined the same way as in the u-AIC. The transition model $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$ describes how the agent transitions from a state $\boldsymbol{x}_t$ to a future state $\boldsymbol{x}_{t+1}$ by taking an action $\boldsymbol{u}_t$. The observation model $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$ for a given sensor, defines the (noisy) relationship between the states and observation.

The general approach for this paper works for any choice of distributions. However, for the sake of simplicity, we choose a Gaussian transition model $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; f(\boldsymbol{x}_t, \boldsymbol{u}_t), \Sigma_f)$, where $f(\cdot)$ is a transition function and Σ_f is the uncertainty over the transition. This is

equivalent to assuming $q \sim \mathcal{N}(\mathbf{0}, \Sigma_f)$ in eq. (1). Similarly, the observation model is chosen to be Gaussian as $p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; g(\boldsymbol{x}_t), \Sigma_y)$.

The unusual element in this model is that the goal state is specified by adding a goal prior over a future state $\boldsymbol{x}_{t+1}$, which we implicitly predict to reach. The optimal control action leading to this can thus be computed by Bayesian inference. The goal prior is also chosen to be Gaussian $\mathcal{N}(\boldsymbol{x}_{t+1}; \boldsymbol{x}_{\text{goal}}, \Sigma_{\text{goal}})$. The smaller the value for Σ_{goal} is, the more aggressively the controller will act to reach the target.

Solving this model requires computing the target posterior $p(\boldsymbol{x}_t, \boldsymbol{u}_t|y_t)$. Computing $p(\boldsymbol{x}_t|y_t)$ results in filtering and computing $p(\boldsymbol{u}_t|y_t)$ results in control. Since all our distributions are Gaussian, the negative log-likelihood of this model will consist of least-square terms and logarithms. For our model this would lead to four terms since we have four Gaussian factors in eq. (5). To simplify notation, we define $||x_1 - x_2||^2_\Sigma = (x_1 - x_2)^\top \Sigma^{-1}(x_1 - x_2) + \ln \Sigma^{-1}$. The full negative log-likelihood for the controller is then:

$$\begin{aligned} -\mathcal{L} = ||\boldsymbol{x}_{t+1} - f(\boldsymbol{x}_t, u_t)||^2_{\Sigma_f} + ||\boldsymbol{y}_t - g(\boldsymbol{x}_t)||^2_{\Sigma_g} + \\ ||\boldsymbol{x}_{t+1} - \boldsymbol{x}_{goal}||^2_{\Sigma_g} + ||\boldsymbol{x}_t - \hat{x}||^2_{\Sigma_1} = F. \end{aligned} \tag{6}$$

We can estimate the state x_t and control action u_t by minimizing the negative log-likelihood by using any suitable optimization method. It follows that the result will depend on the prediction prior ($\hat{x}$) and on the observed value from the sensors $\boldsymbol{y}_t$. Balancing these two terms is equivalent to the prediction and measurement update steps of a Kalman filter, as proved in Ho and Lee (1964). The future state x_{t+1} is constrained by the goal state x_{goal} (third term in $-\mathcal{L}$). Thus, to minimize the whole expression, the value for the control input u_t needs to make the system evolve, according to the model $f(\boldsymbol{x}_t, u_t)$, towards the goal state.

3. FAULT-TOLERANT CONTROL USING PRECISION LEARNING

This section introduces the first major contribution of this paper: achieving FTC with precision learning. Two techniquess will be presented: point-based precision learning, as in (Baioumy et al., 2021b); and a Bayesian method, which is another novel contribution of this paper.

3.1 Point-mass approaches

The observation matrix Σ_y can be updated via gradient descent on $\mathcal{F}$ as: $\dot{\Sigma}_y = -\kappa_\sigma \frac{\partial \mathcal{F}}{\partial \Sigma_y}$. Such update rule has several practical issues. First, in the present case, it would lead to inverting the possibly high dimensional matrix Σ_y. A work around is to directly update its inverse, the *precision* matrix:

$$\dot{\Sigma}_y^{-1} = -\kappa_\sigma \frac{\partial \mathcal{F}}{\partial \Sigma_y^{-1}}. \tag{7}$$

The second issue is that a Σ_y needs to be positive semi-definite. A solution is to perform a re-parameterization with a strictly positive function such as an exponential. An alternative method is to set a lower bound on the variance, as done in Bogacz (2017).

3.2 Bayesian approaches for one-dimensional problems

Once again consider a one-dimensional problem where observations y are affected by noise generated by a Gaussian with a known mean C and scalar precision ω, $\mathcal{N}(y; C, \omega)$. Given some observation y, we wish to find the posterior $p(\omega|y)$. To do this, we choose the prior over the precision $p(\omega)$ to be a *gamma distribution* defined as:

$$\Gamma(\omega; a, b) = \frac{b^a}{\Gamma(a)} \omega^{a-1} e^{-\omega b}$$

where a and b are the parameters of the distribution and $\Gamma(a) = (a-1)!$ is a factorial function. For example, $\Gamma(5) = 4! = 24$. Now to compute the posterior, we multiply the prior with the Gaussian likelihood model of $p(y|\omega)$ and obtain the posterior which is also a gamma distribution as shown below.

$$\begin{aligned} p(\omega) = \Gamma(\omega; a, b) \propto \omega^{a-1} e^{-\omega b} \\ p(\omega|y) \propto p(y|\omega) p(\omega) \propto \omega^{0.5+a-1} e^{-\omega(b + \frac{(y-C)^2}{2})} \\ p(\omega|y) = \Gamma(\omega; a + \frac{1}{2}, b + \frac{(y-C)^2}{2}) \end{aligned}$$

The last equation shows a simple update rule to modify the belief over the precision for every observation. In the optimization for the state, the following quantities are used: expected precision $\mathbb{E}[\omega] = a/b$, $\text{Mode}[\omega] = (a-1)/b$ and $\text{Var}[\omega] = a/b^2$. Note that while a gamma distribution was assumed for the precision, as is common in Bayesian inference, this analysis may extend to other distributions.

3.3 Bayesian approaches for n-dimensional problems

Using the gamma distribution is limited to one-dimensional problems. For n-dimensional problems, the Wishart distribution is used. Wishart distribution is parameterized by $\mathbf{x}$: a squared positive definite matrix of size p containing random variables, and $\mathbf{V}$ a symmetric positive define matrix of size p as well. Then if $n \geq k$ the Wishart distribution over $\mathbf{x}$ is given by:

$$p(\mathbf{x}) = \frac{1}{2^{nk/2} |\mathbf{V}|^{n/2} \Gamma_k\left(\frac{n}{2}\right)} |\mathbf{x}|^{(n-k-1)/2} e^{-(1/2) tr\left(\mathbf{V}^{-1}\mathbf{x}\right)}$$

where $|\mathbf{x}|$ is the determinant of $\mathbf{x}$ and Γ_k is the multivariate gamma function:

$$\Gamma_k\left(\frac{n}{2}\right) = \pi^{k(k-1)/4} \prod_{j=1}^{k} \Gamma\left(\frac{n}{2} - \frac{j-1}{2}\right).$$

Using the Wishart distribution as a prior can then be done in the same fashion as the gamma distribution.

4. INTERPRETABILITY OF PRECISION LEARNING VIA BETA RESIDUALS

This section introduces the second novel contribution of this paper: using the *beta residual* for fault-detection.

The previous section discussed performing FTC using precision learning. While this technique comes with the benefit of fault recovery without the residual thresholds, fault detection was never explicitly performed. Indications that a fault has occurred or is occurring may be useful for the user of the fault-tolerant controller. Common alternative fault-tolerant schemes explicitly detect faults to trigger recovery, providing the user with the controller's belief of the presence of faults. While precision learning does not yield outputs for explicit fault detection, the precision (inverse covariance) of the sensor model is estimated online at every time-step. This section introduces ways to combine explicit fault detection and isolation with precision learning, including a novel approach called the 'beta residual'.

4.1 Precision learning in conjunction with other methods

A straightforward way to achieve fault detection explicitly is to use existing FDI methods in conjunction with precision learning. For instance, we can learn a probabilistically robust threshold from data offline (when the system is not operating) similar to approaches in Baioumy et al. (2021c). Then online (when the system is in operation) we compute the residual signal as $\|y-\hat{x}\|$, where y is the measurement of the sensor and $\hat{x}$ is the estimate of the state x. In our Bayesian Controllers it holds that: $\|y-\hat{x}\| = \|y-\mu_x\|$.

Now fault detection is straightforward to perform: the residual $\|y-\hat{x}\|$ is compared against the learned threshold and a fault is detected once the threshold is exceeded. Fault recovery can happen by triggering precision learning.

This gives us two ways to achieve fault-tolerant control. Precision learning with *implicit fault detection* means that we do not monitor the system and perform precision learning during the whole operation. This means there is no explicit fault detection. Additionally, precision learning with *explicit fault detection* refers to monitoring the systems and only triggering precision learning once a fault is detected. The two methods are compared in the results section and the mean squared error is given in Table 1.

4.2 The beta residual

As discussed, precision learning can work in conjunction with existing fault detection schemes. The controller uses the root mean sqaure error (RMSE) of a Kalman filter $\|y-\hat{x}\|$ as a residual signal and compares it to a learned threshold. We will now consider a different residual based on Bayesian inference, the *beta residual*.

The previous section discusses precision learning as Bayesian inference where instead of computing a point-estimate of the precision, we compute a full distribution. An appropriate choice in this case would be the Gamma distribution, as this is the conjugate prior precision to the Gaussian likelihood noise model with known mean and unknown precision (The Wishart distribution models the precision for the same reason in multi-dimensional cases). This Gamma distribution is parameterised by β and α. Since these parameters encode information about the degree to which a sensor is faulty, one of these parameters may be used as a residual signal. Given that α quantifies the number of observations, it does not correlate with the fault-status of a sensor on its own. Information about the sensor faultiness will be rather encoded in β. The parameter β is inversely proportional to the expected precision:

$$\mathbb{E}[\Gamma(\alpha,\beta)] = \frac{\alpha}{\beta} \tag{8}$$

Now we will discuss how to extract interpretable outputs from the Gamma precision parameter β. Using sensor data labelled with the presence of faults, supervised learning algorithms can be applied to map from β to a probability of a fault. For the sake of demonstration, we use a simple linear classification algorithm, logistic regression Walker and Duncan (1967).

Logistic regression assumes a linear relationship between the log-odds $\ln\frac{p}{1-p}$ and the predictor x: $\ln\frac{p}{1-p} = bx + b_0$, where b and b_0 are model parameters selected to minimize the mean squared error of the classifier. By simple algebraic manipulation, $p = \frac{1}{1+e^{-(bx+b_0)}}$. The sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$ constrains the output of the classifier to always be between 0 and 1. These output prediction are often interpreted as probabilities Walker and Duncan (1967). A logistic regression model can easily be converted to a binary classifier by introducing a threshold on the model's output probability, such as all outputs $p > 0.6$ predicting a fault.

In summary, although precision learning does not explicitly determine the probability of a fault, simulated faults can generate the training data necessary for developing a classifier that uses learned precision statistics to return the probability of a fault. Combined with thresholds on the probability of a fault, the classifier can notify the user of a fault. While precision learning can be triggered using alternative FDI techniques, fault-tolerant control can also be achieved by always using precision learning and extracting interpretable sensor fault detections using beta residuals. This comes with the benefit of having fault classifications align with the controller's belief of the faultiness of sensors: When the sensor is believed to be operating correctly, the inferred precision is high (low variance), while when the sensor is believed to have a fault, the inferred precision will be low (high variance).

5. RESULTS

This section summarizes the results of experimentation in this paper. First, the fault tolerance of the u-AIC with point-mass precision learning is demonstrated. Then, the Bayesian Controller with precision learning as a distribution is shown to outperform alternative model-based FT techniques. Finally, a technique for extracting interpretable outputs from the Bayesian Controller classifying a fault is explained and applied.

5.1 u-AIC with point-mass precision learning

In the first experiment, we apply the methods in Sec. 2.2 on a 2-DOF robotic manipulator. The manipulator has 3 sensor: a position encoder, a velocity encoder and a visual sensor. We use point-based precision learning on the u-AIC.

We test two scenarios: a) precision learning at all times thus performing no explicit fault detection, b) precision learning only when a fault is detected. The first case has no explicit fault detection. In the second case, we use an existing FDI method based on the probabilistically-robust thresholds, then when a fault is detected, precision learning is triggered.

In the simulations, the sensors are injected with zero-mean Gaussian noise. The standard deviation of the noise for encoders is set to $\sigma_q = \sigma_{\dot{q}} = 0.001$, while the one for the camera is set to $\sigma_v = 0.01$. The camera is also affected by barrel distortion with coefficients $K_1 = -1.5e^{-3}$, $K_2 = 5e^{-6}$, $K_3 = 0$ (values are similar to work from Marshall and Lipkin (2014); Piepmeier et al. (2004)). The agent starts in configuration $\boldsymbol{x}_0$, then moves to the targets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. At $t = 8s$ a fault is injected. The encoder fault is such that the output related to the first joint freezes.

Without precision updates, the system is not able to reach the target state after the occurrence of the fault. Instead, the robot arm reaches a different configuration to minimise the free-energy, which is built fusing the sensor information from the (faulty) encoders and the (healthy) camera. However, when the faulty encoder is adjusted using precision learning, the agent is able to reach the final configuration. The Mean Squared Error (MSE) between the belief and the true position ($\boldsymbol{\mu}_x - \boldsymbol{x}$) is computed on a sample of test runs and reported in the Table 1. The results are reported for both the joint whose encoder is faulty, and joints with healthy encoders.

	Joints with encoder fault	Joints without encoder fault
No fault-tolerance	0.0036	0.0020
PL with implicit fault detection	5.422 e-5	4.527 e-5
PL with explicit fault detection	6.097 e-5	4.134 e-5

Table 1. Mean Squared Error (MSE) for different methods of fault-tolerant control. PL indicates precision learning

5.2 The Bayesian controller with precision learning

In the second experiment, we compare precision learning on the Bayesian Controller to existing methods. Extensive simulations are performed on a cruise control to showcase the performance. The agent has two identical sensors, one of which suffers form a fault. We consider 3 types of fault: freezing, sensory injection and sensor drift. The agent is tasked with tracking 3 types of trajectories: constant, linear ramp and sinusoidal.

Benchmarks To evaluate the performance of precision learning, two alternative techniques are applied as benchmarks: (Easy and Fast FDI) EF-FDI Berriri et al. (2012) and (State estimation residual FT) SER FT Kommuri et al. (2016).

In EF-FDI, the detection procedure takes advantage of temporal redundancy in sensor observations, and is chosen here as it is representative of methods that are of a low complexity and fast computationally.

$$r_k = |y_t - 2y_{t-1} + y_{t-2}|,\ R_k = r_k + r_{k-1} + r_{k-2} \tag{9}$$

where y_t is the observation. If the statistic R_k exceeds a threshold δ_{EF}, the sensor output is ignored for the expected duration of the fault.

Similarly, SER FT Kommuri et al. (2016), determines faults using analytical and sensor redundancy. A threshold δSER on the residual for state estimation $|y_t - \hat{x}_t|$ is computed. If this threshold is exceeded, the sensor is deemed faulty and the output is ignored.

5.3 Cruise Control System

The Bayesian controller is tasked with tracking a trajectory for a vehicle cruise control problem given by the following system equations:

$$\boldsymbol{x}_{t+1} = (1 - \frac{b}{m}dt)\boldsymbol{x}_t + \frac{dt}{m}\boldsymbol{u}_t + \boldsymbol{q} \tag{10}$$

where $\boldsymbol{x}_t$ is the velocity of the car at time t, b is the drag coefficient, m is the mass of the car, and $\boldsymbol{u}$ is the control action and $\boldsymbol{q}$ is the process noise. Finally, dt is the timestep size. The discrete observation model when no sensor faults are present is given by

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{\eta} \tag{11}$$

where $\boldsymbol{\eta}$ is the observation noise. The exact parameter values used are provided in Table 2.

Symbol	Name	Value
b	drag coefficient	$5Ns/m$
m	car mass	100 kg
dt	timestep size	0.02 s
$\boldsymbol{q}$	process noise	$\mathcal{N}(0, 0.001m^2s^{-2})$
$\boldsymbol{\eta}$	observation noise	$\mathcal{N}(0, 1m^2s^{-2})$

Table 2. System parameters for the vehicle cruise control problem.

Tracking Performance Applied to the same control problem, each FT technique is tested on 9 different "scenarios" for each FT technique (EF-FDI, No FT, Precision Learning (PL), and SER FT). The scenarios are described below.

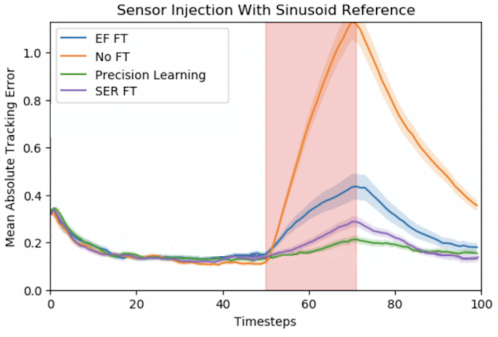


Fig. 1. Mean absolute error in state, where reference is a sinusoid, and the sensor injection is present during the red highlighted period.

To create a quantifiable summary of the relative merit of the different techniques across a range of scenarios, the following procedure was followed. The Bayesian controller is

tasked with a cruise control problem across 900 simulations for each FT technique (EF-FDI, No FT, Precision Learning (PL), and SER FT). The simulations are distributed evenly across sensor fault type (sensor freeze, sensor drift, and sensor injection) and target state trajectory (constant, linear ramp, sinusoidal). Each unique combination of sensor fault type and target state trajectory will be referred to as a 'scenario'. The mean root mean tracking error across 100 experiments for each scenario is compared using heteroskedastic t-tests for difference of means, summarized in Table 3. Each element in the table shows the number of t-tests where the top FT technique's out-performance of the left technique was statistically significant minus the number of t-tests where the reverse was true. Tests were conducted across all three sensor failure types and all three reference paths, for a maximum of 9 in each cell (except for the "Total Merit"). The total Merit entries are the sum of the column, showing total outperformances minus underperformances for a given technique, for a maximum total of 27 (9 per alternative technique). The ensemble mean absolute error across a single scenario, the sensor injection with sinusoid reference is shown in Figure 1.

FT Technique	EF-FDI	No FT	PL	SER FT
EF-FDI	0	-6	8	8
No FT	6	0	8	8
PL	-8	-8	0	-5
SER FT	-8	-8	5	0
Total Merit	-10	-22	21	11

Table 3. Comparison for different FTC methods with the relative merit reported.

5.4 Interpretability using Bayesian residual

In the third experiment, we compare the novel 'beta residual' to the more common State Estimation Residual (equivalent to the RMSE of the Kalman filter).

In the procedure, 1000 sensor faults were generated, with a mean-time-to-fault of 2.8 s, and mean-time-to-recovery of 0.4 s for a total of approximately 160,000 simulation timesteps. The occurrences of the faults and recoveries are distributed assuming a constant probability of failure. The faults are therefore Poisson-distributed in their time-to-occurence and in length. When a fault is triggered in the simulation, a random selection of one of the 3 faults is selected. The model assumes that only one fault at a time occurs with the same sensor. Faults are allowed to occur simultaneously at both of the sensors (although this is not common). The learned precision parameters and the SER residual are saved together with the binary fault truth values. The results are split into 80% training set and 20% test set. Logistic regression is trained on the training set and evaluated on the test set.

The test set Receiver Operating Characteristic (ROC) is shown in Table 4. The ROC-AUC is the are under the ROC. It is a common measure of the performance of binary classifiers that return probabilities rather than classes. For the controller with the given FT strategy on the left, the model input is used to predict the 0-1 faultiness of the sensor with logistic regression. The ROC-AUC, an assessment of the classification power of the model.

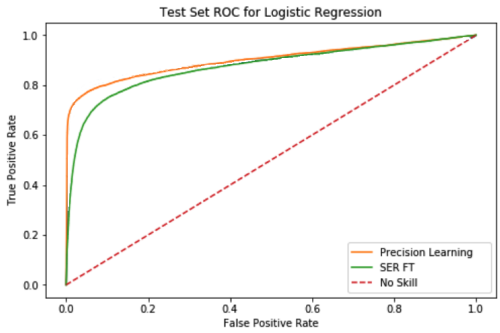


Fig. 2. The Receiver Operating Characteristic (ROC) shows the trade-off between true positive rate and false positive rate for all possible thresholds. The diagonal line in red is the ROC of a hypothetical model that guesses randomly. EF-FFT is not shown for comparison because its residual for fault detection corresponds with the onset of faults, not the presence of faults, rendering it unsuitable for classification using logistic regression.

A brief explanation of the ROC as a metric follows. In order to evaluate a classifier, accuracy is an insufficient metric, due to imbalances in the classes (there are more non-faulty cases than faulty cases). In the case of binary classification, both the true positive rate and the false positive rate should be considered. However, in order to evaluate the logistic regression model as a classifier, a probability threshold for classification must be selected. The ROC evaluates the model at all possible thresholds: Each point on the curve represents a threshold with an associated false positive rate (x-axis) and true positive rate (y-axis). The area under the curve measures the fit of the logistic regression model: The closer to 1, the better the classifier. A random guesser has an ROC-AUC (Area Under Curve) of 0.5, labelled as "No Skill" in Figure 2.

These results show that the parameter beta can be effectively used as a residual signal. A stochastic threshold was learned using logistic regression, however, other supervised models may be used.

FT Technique	Input	AUC
SER FT	$\|y - \hat{x}\|$	0.873
Precision Learning	β	0.907

Table 4. ROC-AUC for fault detection with different residuals.

6. DISCUSSION AND CONCLUSION

This paper compares a set of fault-tolerant controllers based on Bayesian inference and on a novel approach: precision learning. All controllers perform state-estimation, control, fault detection and recovery as a single inference procedure. Sensor faults are modelled as changes in the covariance/precision of the sensor model. Thus by learning the precision online, the agent can achieve fault-tolerant control. In addition, by modeling the control problem as Bayesian inference, standard methods using non-Gaussian distributions and non-linear systems can be leveraged. The results show how controllers based on precision learning outperform existing approaches on a variety of tasks. This

includes different types of fault (sensor freeze, sensor injection and drift), while tracking different trajectories (ramp, step-response, and sinusoid).

A key limitation of this approach is the lack of an explicit fault detection step. To mitigate this issue, one could use standard fault detection approaches in conjunction with precision learning. We demonstrate that this yield satisfactory performance. Additionally, we introduce a novel *beta residual*, which shows improved performance compared the commonly used residuals such as the root mean square error (RMSE) of a Kalman filter. Additionally, it is pointed out how the beta residual can be interpreted as an hyperparameter which represents the trust in a sensor at a specific time instant. Future work will further investigate the use of the beta residual for fault-detection, including its fault detectability properties.

REFERENCES

Baioumy, M., Duckworth, P., Lacerda, B., and Hawes, N. (2021a). Active inference for integrated state-estimation, control, and learning. In *Proc of IEEE Int. Conf. on robotics and automation (ICRA)*.

Baioumy, M., Mattamala, M., and Hawes, N. (2020). Variational inference for predictive and reactive controllers. In *ICRA 2020 Workshop on New advances in Brain-inspired Perception, Interaction and Learning*. Paris, France.

Baioumy, M., Pezzato, C., Corbato, C.H., Hawes, N., and Ferrari, R. (2021b). Towards stochastic fault-tolerant control usingprecision learning and active inference. In *Int. Workshop on Active Inference*.

Baioumy, M., Pezzato, C., Ferrari, R., Corbato, C.H., and Hawes, N. (2021c). Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference. In *European Control Conf. (ECC)*.

Berriri, H., Naouar, M.W., and Slama-Belkhodja, I. (2012). Easy and fast sensor fault detection and isolation algorithm for electrical drives. *IEEE Trans. on Power Electronics*, 27(2), 490–499. doi:10.1109/TPEL.2011.2140333.

Blanke, M., Frei, W.C., Kraus, F., Patton, J.R., and Staroswiecki, M. (2000). What is fault-tolerant control? *IFAC Proceedings Volumes*, 33(11), 41–52.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76, 198–211.

Buckley, C.L., Kim, C.S., McGregor, S., and Seth, A.K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81, 55–79.

Cai, B., Huang, L., and Xie, M. (2017). Bayesian networks in fault diagnosis. *IEEE Trans. Ind. Inf.*, 13(5), 2227–2240.

Chen, J. and Patton, R.J. (1999). *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, LLC.

Codetta-Raiteri, D. and Portinale, L. (2015). Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *IEEE Trans. Syst. Man Cybern.*, 45(1), 13–24.

Fox, C.W. and Roberts, S.J. (2012). A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2), 85–95.

Friston, K.J. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2), 27–138.

Friston, K., Stephan, K., Li, B., and Daunizeau, J. (2010). Generalised filtering. *Mathematical Problems in Engineering*.

Gao, Z., Cecati, C., and Ding, S.X. (2015). A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Trans. on industrial electronics*, 62(6), 3757–3767.

Ho, Y. and Lee, R. (1964). A bayesian approach to problems in stochastic estimation and control. *IEEE Trans. on automatic control*, 9(4), 333–339.

Kommuri, S.K., Defoort, M., Karimi, H.R., and Veluvolu, K.C. (2016). A robust observer-based sensor fault-tolerant control for pmsm in electric vehicles. *IEEE Trans. on Industrial Electronics*, 63(12), 7671–7681.

Lanillos, P. and Cheng, G. (2018). Adaptive robot body learning and estimation through predictive coding. In *IROS*.

Lerner, U., Parr, R., Koller, D., Biswas, G., and Others (2000). Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI*, 531–537. aaai.org.

Marshall, M. and Lipkin, H. (2014). Kalman filtering visual servoing control law. In *IEEE Procs. of Int. Conf. on Mechatronics and Automation*.

Murphy, K.P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Pezzato, C., Ferrari, R., and Corbato, C.H. (2020). A novel adaptive controller for robot manipulators based on active inference. *IEEE Robotics and Automation Letters*, 5(2), 2973–2980.

Pezzato, C., Baioumy, M., Corbato, C.H., Hawes, N., Wisse, M., and Ferrari, R. (2020). Active inference for fault tolerant control of robot manipulators with sensory faults. In Springer (ed.), *1st Int. Workshop on Active Inference, ECML PKDD*, volume 1326 of *Communications in Computer and Information Science*.

Piepmeier, J., McMurray, G., and Lipkin, H. (2004). Uncalibrated dynamic visual servoing. In *IEEE Trans. on Robotics and Automation*, volume 20, 143–147.

Sun, W., Paiva, A.R.C., Xu, P., Sundaram, A., and Braatz, R.D. (2020). Fault detection and identification using Bayesian recurrent neural networks. *Comput. Chem. Eng.*, 141, 106991.

Verma, V., Gordon, G., Simmons, R., and Thrun, S. (2004). Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robot. Autom. Mag.*, 11(2), 56–66.

Walker, S.H. and Duncan, D.B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2), 167–179.

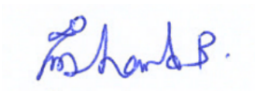
Statement of Authorship for joint/multi-authored papers for PGR thesis

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

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

Title of Paper	Beta Residuals: Improving Fault Detection and Recovery via Bayesian Inference and Precision Learning
Publication Status	Published
Publication Details	Mohamed Baioumy, William Hartenmink, Riccardo Ferrari, Nick Hawes, "Beta Residuals: Improving Fault Detection and Recovery via Bayesian Inference and Precision Learning", 11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS 2022

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	I led the development of the idea in collaboration with Riccardo Ferrari. I performed the literature review with assistance from William Hartenmind and Riccardo Ferrari. I implemented the initial algorithm and carried out the experiments in section 5.1. William Hartenmink carried out the rest of the experiments and analyzed the results. William's work was supervised by me and Nick Hawes. The paper was written by me with support from William Hartenmink and Riccardo Ferrari.		
Signature 		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

V. Planning under uncertainty

A. *On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference*

This paper presents a new method for solving the Stochastic Shortest-Path Markov Decision Process (SSP MDP) with indefinite horizon through probabilistic inference. We explore the distinctions between dynamic programming techniques traditionally used in artificial intelligence for solving MDPs and approaches employed in active inference. The key contributions of this work are: 1. An algorithm for solving SSP MDPs using probabilistic inference, accommodating indefinite horizon scenarios. 2. An in-depth comparison between MDP solution strategies in the AI community and active inference.

Now we discuss the potential utility of utilizing probabilistic inference in this paper.

- **Novel Capabilities (×):** In this paper, we formulate an SSP MDP as a probabilistic inference problem and demonstrate its equivalence to Policy Iteration. Thus, no novel capabilities are introduced. We simply show an equivalent method using a different paradigm.
- **Coherence (✓):** Here we only consider a planning problem, fully presented as a probabilistic inference problem. Technically coherence applies, however it is not significant in this context as there is only one problem being solved.
- **Joint Optimization (×) :** This is not applicable as there is only one problem being solved.
- **Automated Solvers (✓):** We can rely on a standard message passing algorithm to perform inference in this case. This is the main benefit of framing an SSP MDP as probabilistic inference. In the case of a fully observable discrete problem, this is exactly equivalent to standard policy iteration. However, if one wishes to extend this approach to continuous or partially observable MDPs, the probabilistic inference perspective is immensely helpful. Using an automated solved for probabilistic inference, one can change a few lines of code and solve complex variants of the SSP MDP directly. This is demonstrated in Section V-C.
- **Connection to Literature (✓) :** We connect planning and policy search approaches and highlight their connection to inference-based methods.

The paper starts on the following page.

On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference

Mohamed Baioumy, Bruno Lacerda, Paul Duckworth, and Nick Hawes

Oxford Robotics Institute, University of Oxford
{mohamed, bruno, pduckworth, nickh}@robots.ox.ac.uk

Abstract. Previous work on planning as active inference addresses finite horizon problems and solutions valid for *online* planning. We propose solving the general Stochastic Shortest-Path Markov Decision Process (SSP MDP) as probabilistic inference. Furthermore, we discuss online and offline methods for planning under uncertainty. In an SSP MDP, the horizon is *indefinite* and unknown a priori. SSP MDPs generalize finite and infinite horizon MDPs and are widely used in the artificial intelligence community. Additionally, we highlight some of the differences between solving an MDP using dynamic programming approaches widely used in the artificial intelligence community and approaches used in the active inference community. F

1 Introduction

A core problem in the field of artificial intelligence (AI) is building agents capable of automated planning under uncertainty. Problems involving planning under uncertainty are typically formulated as an instance of a Markov Decision Process (MDP). At a high level, an MDP comprises 1) a set of world states, 2) a set of actions, 3) a transition model describing the probability of transitioning to a new state when taking an action in the current state, and 4) an objective function (e.g. minimizing costs over a sequence of time steps). An MDP solution determines the agent's actions at each decision point. An optimal MDP solution is one that optimizes the objective function. These are typically obtained using dynamic programming algorithms [1] [17,26].

Recent work based on the active inference framework [13] poses the planning problem as a probabilistic inference problem. Several papers have been published showing connections between active inference and dynamic programming to solve an MDP [15,8,7]. However, the planning problem being solved in the two communities is not equivalent.

First, dynamic programming approaches used to solve an MDP, such as policy iteration, are valid for finite, infinite and indefinite horizons. Indefinite horizons are finite but of which the length is unknown a priori. For instance, consider

[1] Linear programming approaches are also popular methods for solving MDPs [2,12,22,9]. Additionally, other methods exist in the reinforcement learning community such as policy gradient methods [28,14,27].

an agent navigating from a starting state to a goal state in a grid world where the outcome is uncertain (e.g. the 4×4 grid world in Figure 1 shown in the appendix). Before starting to act in the environment, there is no way for the agent to know how many time steps it will take to reach the goal. Algorithms based on dynamic programming, such as policy iteration, are valid for such settings. They can solve the Stochastic Shortest-Path Markov decision process (SSP MDP)[17,2]. However, work from active inference is only formulated for finite horizons [8].

Second, the optimal solution to an SSP MDP is a stationary deterministic policy [17]. This refers to a mapping from states to actions independent of time. Computing this optimal policy can be done *offline* (without interaction with the environment) or online (while interacting). In the active inference literature however, solving the planning problem is performed by computing a stochastic plan (a sequence of actions given the current state). This is only valid during online planning. Additionally, the solution is only optimal given a certain horizon, which is specified a priori. If the horizon chosen is too short, the agent will not find a solution to reach the target. If it is too long, the solution will be sub-optimal.

The main contribution of this paper is presenting a novel algorithm to solve a Stochastic Shortest-Path Markov Decision Process using probabilistic inference. This is an MDP with an *indefinite horizon*. Additionally, highlighting the several gaps between solving an MDP in the AI community and the active inference community.

Section 2 discusses the SSP MDP and section 3 presets an approach for solving an SSP MDP as probabilistic inference. The equivalence between the two methods is shown in Section 4.1. Furthermore, the difference between world states and temporal state is highlighted in Section 4.2. Policies, plans and probabilistic plans are discussed in Section 4.3. Finally, a discussion on online vs offline planning can be found in Section 5.

2 Stochastic Shortest Path MDP

An SSP MDP is defined as a tuple $\mathcal{M} = (S, A, C, T, G)$. S is the set of states, A is the set of actions, $C : S \times A \times S \to \mathbb{R}$ is the cost function, and $T : S \times A \times S \to [0, 1]$ is the transition function. $G \subset S$ is the set of goal states. Each goal state $s_g \in G$ is absorbing and incurs zero cost. The expected cost of applying action a in state s is $\bar{C}(s, a) = \sum_{s' \in S} T(s, a, s') \cdot C(s, a, s')$. The minimum expected cost at state s is $\bar{C}^*(s) = \min_{a \in A} \bar{C}(s, a)$. A policy maps a state to a distribution over action choices. A policy is deterministic if it chooses a single action at each step. A policy π is proper if it reaches $s_g \in G$ starting from any s with probability 1. In an SSP MDP, the following assumptions are made [17]: a) there exists a proper policy, and b) every improper policy incurs infinite cost at all states where it is improper.

The goal is to find the an *optimal policy* π^* with the minimum expected cost $\bar{C}^*(s)$ and can be computed as

$$\pi^*(s) = \text{argmin}_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, a, s'\right) \left[\mathcal{C}\left(s, a, s'\right) + V^*\left(s'\right) \right] \right].$$

$V^*(s)$ is referred to as the optimal value for a state s and is defined as:

$$V^*(s) = \min_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, a, s'\right) \left[\mathcal{C}\left(s, a, s'\right) + V^*\left(s'\right) \right] \right].$$

Crucially, the optimal policy π^* corresponding to the optimal value function is Markovian (only dependant on the current state) and deterministic [17]. Solving an SSP MDP means finding a policy that minimizes expected cost, as opposed to one that maximizes reward. This difference is purely semantic as the problems are dual. We can define a reward function $\mathcal{R} = -\mathcal{C}$ and move to a reward maximization formulation. A more fundamental distinction is the presence of a special set of (terminal) goal states, in which staying forever incurs no cost.

Solving an SSP MDP can be done using standard dynamic programming algorithms such as policy iteration. Policy iteration can be divided in two steps, policy evaluation and improvement. In policy evaluation, for a policy π, the value function $V_\pi(s)$ is recursively evaluated until convergence as

$$V_\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, \pi(s), s'\right) \left[\mathcal{C}\left(s, \pi(s), s'\right) + V_\pi\left(s'\right) \right].$$

In the policy improvement step, the state-action value function $Q(s, a)$ is computed as:

$$Q(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') [\mathcal{C}(s, a, s') + V(s')].$$

Then we compute a new policy as $\pi' = \text{argmin}_{a \in \mathcal{A}} Q(s, a)$ for every state in S. Iterating between these two steps guarantees convergence to an optimal policy.

Properties of an SSP MDP. An SSP MDP can be shown to generalize finite, infinite and indefinite horizon MDPs [3,17]. Thus algorithms valid for an SSP MDP are also valid for the finite and infinite horizon MDPs. Additionally, it can be proven that each SSP MDP has an optimal deterministic policy independent of time. Therefore, the claims made in [8] about active inference being more general since it computes stochastic policies are unjustified when solving an MDP. However, these results do not hold in partially observable cases or in the presence of uncertain models. But in the SSP MDP defined above (which is commonly used in the AI community), there always exists a deterministic optimal policy. Note that there is an infinite number of stochastic policies but only a finite number of deterministic policies ($|\mathcal{S}|^{|\mathcal{A}|}$). This greatly speeds up the algorithms while still guaranteeing optimality.

3 Solving an SSP MDP as probabilistic inference

In this section we discuss a novel approach for solving an SSP MDP as probabilistic inference. We use an inference algorithm that exactly solves an SSP MDP as defined in the previous section. This approach is inspired by work from [32,31] which solves an MDP with an indefinite horizon. This approach has been successfully applied to solve problems of planning under uncertainty, e.g. [18,30].

3.1 Definitions

The definition of an SSP MDP includes a set of world states $\mathcal{S}$ and actions $\mathcal{A}$. In probabilistic inference we instead reason about *temporal* states and actions. A temporal state s_t is a random variable defined over all world states. Conceptually it represents the state that the agent will visit at the time-step t. For the grid world in Figure 1, there are 16 world states but the number of temporal states is unknown a priori since the horizon is unknown.

The transition probability is defined as a probability distribution over temporal states and actions as $P(s_{t+1}|a_t, s_t)$. If the random variables are fixed to specific world states i and j and an action a, the transition probability $P(s_{t+1} = j|a_t = a, s_t = i)$ would be equivalent to the transition function $\mathcal{T}$ defined for an SSP MDP. The probability of taking a certain action in a state is parameterized by a policy π as $P(a_t = a|s_t = i; \pi) = \pi_{ai}$. This policy is defined exactly the same as in the case of an SSP MDP.

The cost function $P(c_t|s_t, a_t)$ is defined differently. The temporal cost variables c_t are defined as binary random variables $c_t \in \{0, 1\}$. Translating an arbitrary cost function to temporal costs can be done by scaling the cost function $\mathcal{C}(s, a)$ (as defined in the previous section) between the minimum cost ($\min(\mathcal{C})$) and maximum cost ($\max(\mathcal{C})$) as:

$$P(c_t = 1 \mid a_t = a, s_t = s) = \frac{\mathcal{C}(a, s) - \min(\mathcal{C})}{\max(\mathcal{C}) - \min(\mathcal{C})}.$$

Any expression with $P(c_t = 1)$ can be though of as 'the probability of a cost being maximal'. Thus, the probability of a cost being maximal given a state and an action is $P(c_t = 1 \mid a_t = a, s_t = s)$. Now we can reason about the highest possible cost for a state s and action a as one where $P(c_t = 1 \mid a_t = a, s_t = s) = 1$ and the lowest possible cost as $P(c_t = 1 \mid a_t = a, s_t = s) = 0$. Any other cost will have a probability in-between, according to its magnitude.

Finally, we model the horizon as a random variable. The temporal states and actions are considered up to the end of the horizon T. However, the horizon is generally unknown. We thus model T itself as a random variable. Combining all this information we can define the SSP MDP using a probabilistic model.

3.2 Mixture of finite MDPs

In this section we define the SSP MDP in terms of a mixture of finite MDPs with only a final cost variable. Given every horizon (for instance $T = 1$) the

finite MDP can be given as $P\left(c, s_{0:T}, a_{0:T} \mid T; \boldsymbol{\pi}\right)$. Note that we dropped the time-index for c_t since there is only one cost variable now. This model can be factorized as

$$P\left(c, s_{0:T}, a_{0:T} \mid T; \boldsymbol{\pi}\right) = P\left(c \mid a_T, s_T\right) P\left(a_0 \mid s_0; \boldsymbol{\pi}\right) \\ P\left(s_0\right) \cdot \prod_{t=1}^{T} P\left(a_t \mid s_t; \boldsymbol{\pi}\right) P\left(s_t \mid a_{t-1}, s_{t-1}\right)$$

To reason about the full MDP, we consider the mixture model of the joint given by the joint probability distribution

$$P\left(c, s_{0:T}, a_{0:T}, T; \boldsymbol{\pi}\right) = P\left(c, s_{0:T}, a_{0:T} \mid T; \boldsymbol{\pi}\right) P(T)$$

where $P(T)$ is a prior over the total time, which we choose to be a flat prior (uniform distribution).

3.3 Computing an optimal policy

Our objective is to find a policy that minimizes the expected cost. Similarly to policy iteration, we do not assume any knowledge about the initial state. Expectation-Maximization[2] can be used to find the optimal parameters of our model: the policy π. The E-step will, for a given π, compute a posterior over state-action sequences. The M-step then adapts the model parameters π to optimize the expected likelihood with respect to the quantities calculated in the E-step.

E-step: a backwards pass in all finite MDPs. We use the simpler notation $p(j \mid a, i) \equiv P\left(s_{t+1} = j \mid a_t = a, s_t = i\right)$ and $p(j \mid i; \boldsymbol{\pi}) \equiv P\left(s_{t+1} = j \mid s_t = i; \boldsymbol{\pi}\right) = \sum_a p(j \mid a, i)\pi_{ai}$. Further, as a 'base' for backward propagation, we define

$$\beta_0(i) = P\left(c = 1 \mid x_T = i; \boldsymbol{\pi}\right) = \sum_a P\left(c = 1 \mid a_T = a, x_T = i\right) \pi_{ai}.$$

This is the immediate cost when following a policy π. It is the expected cost if there is only one time-step remaining. Then, we can recursively compute all the other backward messages. We use the index τ to indicate a backwards counter. This means that $\tau + t = T$, where T is total (unknown) horizon length. This is computed as

$$\beta_\tau(i) = P\left(c = 1 \mid x_{T-\tau} = i; \boldsymbol{\pi}\right) = \sum_j p(j \mid i; \boldsymbol{\pi}) \beta_{\tau-1}(j).$$

Intuitively, the backward messages are the expected cost if one incurs a cost at the last time step only. So, β_2, is the expected cost if the agent follows the policy π for two time-steps and only incurs a cost after that. Using these messages, we can compute a value function dependent on time, actions and states given as:

[2] An expectation-maximization algorithm can be viewed as performing free-energy minimization [21,16]. In the E-step, the free-energy is computed and the M-step updates the parameters to minimize the free-energy.

$$q_\tau(a,i) = P\left(c=1 \mid a_t=a, s_t=i, T=t+\tau; \boldsymbol{\pi}\right) = \begin{cases} \sum_j p(j \mid i,a)\beta_{\tau-1}(j) & \tau > 1 \\ P\left(c=1 \mid a_T=a, s_T=i\right) & \tau = 0. \end{cases}$$

Marginalizing out time, we get the state-action value-function

$$P\left(c=1 \mid a_t=a, s_t=i; \boldsymbol{\pi}\right) = \frac{1}{C}\sum_\tau P(T=t+\tau) q_\tau(a,i)$$

where C is a normalization constant. This quantity is the probability of getting a maximum cost given a state and action. It is similar to the $Q(s,a)$ function computed in policy iteration.

M-step: the policy improvement step. The standard M-step in an EM-algorithm maximizes the expected complete log-likelihood with respect to the new parameters π'. Given that the optimal policy for an MDP is deterministic, a greedy M-step can be used. However, our goal is to minimize the log-likelihood in this case as it refers to a the probability of receiving a maximal cost. This can done as

$$\pi' = \underset{a}{\operatorname{argmin}}(P\left(c=1 \mid a_t=a, s_t=i; \boldsymbol{\pi}\right)) \tag{1}$$

This update converges much faster than in a standard M-step. Here an M-step can be used to to obtain a stochastic policy. However, this is unnecessary since the optimal policy is deterministic. Note that there is an infinite number of stochastic policies but a finite number of deterministic ones. In conclusion, a greedy M-step is faster to converge but still guarantees an optimal policy.

4 Connections between the two views

4.1 Exact relationship between policy iteration and planning as probabilistic inference

The messages β computed during backward propagation are exactly equal to the value functions for a single MDP of finite time. The full value function is can therefore be written as the sum of the βs,

$$V_\pi(i) = \sum_T \beta_T(i)$$

since the prior over time $P(T)$ is a uniform prior. If $P(T)$ is not a uniform distribution, this would result in a mixture rather than a sum. The same applies to the relationship between the Q-value function:

$$Q_\pi(a,i) = \sum_T q_T(a,i).$$

Hence, the E-step essentially performs a policy evaluation which yields the classical value function. Given this relation to policy evaluation, the M-step performs an operation exactly equivalent to standard policy improvement. Thus, the EM-algorithm using exact inference is equivalent to Policy Iteration but computes the necessary quantities differently.

One unanswered question is when to stop computing the backward messages. In [32] messages are computed up to a number T_{max}. From this perspective, the planning as inference algorithm presented is equivalent to the so-called *truncated policy iteration* algorithm as opposed to the more common ϵ-*greedy* version.

In the policy evaluation step, one iterates through the state space to update the value $v_\pi(s)$ for every state until a termination criterion is met. An ϵ-greedy criterion means that we stop iterating though the state space once the maximum difference in $V_\pi(s)$ for any s is smaller than a positive small number ϵ. In truncated policy iteration, however, we iterate through the state space $T_{\max}$ times and then perform the policy improvement step. The probabilistic inference algorithm presented in this paper is equivalent to truncated policy iteration if we restrict the maximum number of β messages to be computed.

4.2 World states vs temporal states

In dynamic programming, one reasons over the world states. In the grid world example in Figure 1, this refers to a grid cell. This grid world has 16 world states. In probabilistic inference, one reasons about a *temporal state*. This is a random variable over all world states. The number of temporal states is dependent on how many time-steps the agents acts in the environment (which is often unknown beforehand). An illustration of the difference is given in Fig. 2.

4.3 Policies, plans and probabilistic plans

A classical planning algorithm computes a plan: a sequence of actions. An algorithm like A* or Dijkstra's algorithm [5] can be used to find the optimal path from a stating state to a goal state, given a deterministic world. Crucially, this solution can be computed offline (without interactions with the environment). In a stochastic world, this does not work since the agent can not predict in which states it will end up. However, one can use deterministic planning algorithms for stochastic environments if the path is re-planned online at every time-step. Determinization-based methods have found success in solving planning under uncertainity problems such as the famous FF-replan algorithm [34]

Active inference approaches computes a *probabilistic plan*. The active inference literature calls this a policy; however, we use a different term to avoid confusion.[3] In active inference, the agents computes a finite plan while interacting with the environment. However, rather than assuming a deterministic world

[3] The distinction between a plan and a policy when using active inference has been briefly discussed in [20]. Additionally, other methods computing plans as probabilistic inference have been proposed before active inference in [1,33]

(like FF-replan [34]), the probabilities are taken into account. This can be shown to compute the optimal solution to an MDP (when planning online). We thus refer to it as a probabilistic plan, a plan that was computed while taking the transition probabilities into account.

Finally, a policy is a mapping from states to actions, i.e. the agent has a preferred action to take for every state. Policies can be stochastic or time-dependent; however, for an SSP MDP the optimal policy is deterministic and independent of time. An agent can compute a policy offline and use it online without needing any additional computation while interacting with the environment. The difference between a plan and policy is illustrated in Fig. 1.

To summarize, a plan or a probabilistic plan can only be used for online planning. Since the outcome of an action is inherently uncertain. Probabilistic plans (as used in active inference) find an optimal solution when used to plan online. A policy also provides an optimal solution and can be computed offline or online.

5 Discussion

In this paper we present a novel approach to solve a stochastic shortest path Markov decision process (SSP MDP) as probabilistic inference. The SSP MDP generalizes many models, including finite and infinite MDPs. Crucially, the dynamic programming algorithms (such as policy iteration) classically used to solve an SSP MDP are valid for *indefinite horizons* (finite but of unknown length); this is not the case for active inference approaches.

The exact connections between solving an MDP using policy iteration and the presented algorithm are discussed. Afterwards, we discussed the gap between solving an MDP in active inference and the approaches in the artificial intelligence community. This included the difference between world states and temporal states, the difference between plans, probabilistic plans and policies. An interesting question now is, which approach is more appropriate? This depends on the problem at hand and whether it can be solved online or offline.

Online and offline planning. As discussed in Section 4.3, a policy is mapping from states to actions and can be used for offline and online planning. Computing a policy is somewhat computationally expensive; however, a look-up is very cheap. Thus if one operates in an environment where the transition and cost function do not change, it is best to compute an optimal policy offline then use it online (while interacting with the environment). This is the case for many planning and scheduling problems, such as a set of elevators operating in sync [6], task-level planning in robotics [19], multi-objective planning [23,11] and playing games [4,25]. The challenges in these problems are often that the state-space is incredibly large and thus approximations are needed. However, the problem is fully observable and the cost and transition models are static; the rules of chess do not change half way, for instance.

If the transition or cost functions vary while interacting with the environment (e.g. [24,10]), an offline solution is not optimal. In this case, the agent can plan online by re-evaluating a policy or computing probabilistic plans (as done in active inference). Computing the latter is cheaper and requires less memory. This is because a probabilistic plan is a distribution over actions $p(a_t)$ up to a time horizon T while a (finite policy) is a conditional distribution $p(a_t|s_t)$ over all world states in S. For any time-step, the posterior over the action is related to the policy such that $p(a_t) = \sum_s p(a_t|s_t)p(s_t)$.

Consider the work in [29,24]. In both cases a robot operates in an environment susceptible to changes. If the environment changes, the agent can easily construct a new model by varying the cost or transition function but needs to recompute a solution. In [29] the authors recompute a policy at every time-step while in [24] a probabilistic plans is computed using active inference. Since in both cases the solution is recomputed at every time-step, active inference is preferred since it requires less memory and can be computationally cheaper. On the other hand, if the environment only changes occasionally, computing a policy might remain preferable.

To conclude, if the transition and cost functions ($\mathcal{T}$ and $\mathcal{C}$) are static, it is preferable to compute a policy offline. If $\mathcal{T}$ and $\mathcal{C}$ change occasionally, one may still compute an offline policy and recompute a policy only when a change occurs. However, if the environment is dynamic, computing a probabilistic plan (using active inference) is preferable to recomputing a policy at every time-step.

References

1. Attias, H.: Planning by probabilistic inference. In: AISTATS (2003)
2. Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Mathematics of Operations Research **16**(3), 580–595 (1991)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming: an overview. In: Proceedings of 1995 34th IEEE conference on decision and control. vol. 1, pp. 560–564. IEEE (1995)
4. Campbell, M., Hoane, A.J., Hsu, F.: Deep blue. Artif. Intell. **134**, 57–83 (2002)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT press (2009)
6. Crites, R.H., Barto, A.G., et al.: Improving elevator performance using reinforcement learning. Advances in neural information processing systems pp. 1017–1023 (1996)
7. Da Costa, L., Parr, T., Sajid, N., Veselic, S., Neacsu, V., Friston, K.: Active inference on discrete state-spaces: a synthesis. arXiv preprint arXiv:2001.07203 (2020)
8. Da Costa, L., Sajid, N., Parr, T., Friston, K., Smith, R.: The relationship between dynamic programming and active inference: The discrete, finite-horizon case. arXiv preprint arXiv:2009.08111 (2020)
9. d'Epenoux, F.: A probabilistic production and inventory problem. Management Science **10**(1), 98–108 (1963)
10. Duckworth, P., Lacerda, B., Hawes, N.: Time-bounded mission planning in time-varying domains with semi-mdps and gaussian processes (2021)

11. Etessami, K., Kwiatkowska, M., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of markov decision processes. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 50–65. Springer (2007)
12. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: International school on formal methods for the design of computer, communication and software systems. pp. 53–113. Springer (2011)
13. Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., Pezzulo, G.: Active inference: a process theory. Neural computation **29**(1), 1–49 (2017)
14. Grondman, I., Busoniu, L., Lopes, G.A., Babuska, R.: A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**(6), 1291–1307 (2012)
15. Kaplan, R., Friston, K.J.: Planning and navigation as active inference. Biological cybernetics **112**(4), 323–343 (2018)
16. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
17. Kolobov, A.: Planning with Markov Decision Processes: An AI Perspective, vol. 6. Morgan & Claypool Publishers (2012)
18. Kumar, A., Zilberstein, S., Toussaint, M.: Probabilistic inference techniques for scalable multiagent decision making. Journal of Artificial Intelligence Research **53**, 223–270 (2015)
19. Lacerda, B., Faruq, F., Parker, D., Hawes, N.: Probabilistic planning with formal performance guarantees for mobile service robots. The International Journal of Robotics Research **38**(9), 1098–1123 (2019)
20. Millidge, B., Tschantz, A., Seth, A.K., Buckley, C.L.: On the relationship between active inference and control as inference. In: International Workshop on Active Inference. pp. 3–11. Springer (2020)
21. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
22. Nazareth, J.L., Kulkarni, R.B.: Linear programming formulations of markov decision processes. Operations research letters **5**(1), 13–16 (1986)
23. Painter, M., Lacerda, B., Hawes, N.: Convex hull monte-carlo tree-search. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 30, pp. 217–225 (2020)
24. Pezzato, C., Hernandez, C., Wisse, M.: Active inference and behavior trees for reactive action planning and execution in robotics. arXiv preprint arXiv:2011.09756 (2020)
25. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)
26. Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning, vol. 135. MIT press Cambridge (1998)
27. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems. pp. 1057–1063 (2000)
28. Thomas, P.S., Brunskill, E.: Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. arXiv preprint arXiv:1706.06643 (2017)

29. Tomy, M., Lacerda, B., Hawes, N., Wyatt, J.L.: Battery charge scheduling in long-life autonomous mobile robots via multi-objective decision making under uncertainty. Robotics and Autonomous Systems **133**, 103629 (2020)
30. Toussaint, M., Charlin, L., Poupart, P.: Hierarchical pomdp controller optimization by likelihood maximization. In: UAI. vol. 24, pp. 562–570 (2008)
31. Toussaint, M., Harmeling, S., Storkey, A.: Probabilistic inference for solving (po) mdps. University of Edinburgh, School of Informatics Research Report EDI-INF-RR-0934 (2006)
32. Toussaint, M., Storkey, A.: Probabilistic inference for solving discrete and continuous state markov decision processes. In: Proceedings of the 23rd international conference on Machine learning. pp. 945–952. ACM (2006)
33. Verma, D., Rao, R.P.: Goal-based imitation as probabilistic inference over graphical models. In: Advances in neural information processing systems. pp. 1393–1400 (2006)
34. Yoon, S.W., Fern, A., Givan, R.: Ff-replan: A baseline for probabilistic planning. In: ICAPS. vol. 7, pp. 352–359 (2007)

A Appendix: Illustrations

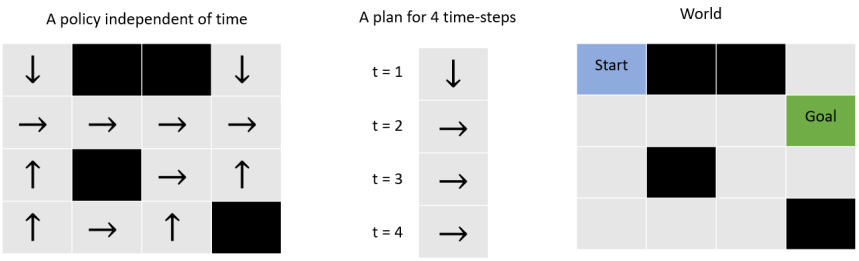


Fig. 1. An illustration of a 4×4 grid world **(right)**. The initial state is blue and goal state is green. An illustration for a policy (**left**) and a plan (**middle**).

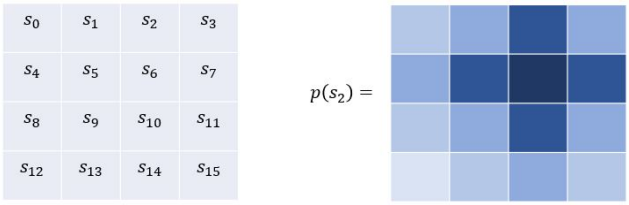


Fig. 2. Annotated world states (**left**) and a posterior over a temporal state (**right**).

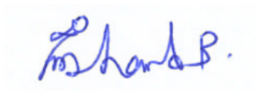
Statement of Authorship for joint/multi-authored papers for PGR thesis

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

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

Title of Paper	On Solving a Stochastic Shortest-path Markov Decision Process as Probabilistic Inference
Publication Status	Published
Publication Details	Baioumy, Mohamed, Bruno Lacerda, Paul Duckworth, and Nick Hawes. "On solving a stochastic shortest-path Markov decision process as probabilistic inference." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 819-829. Cham: Springer International Publishing, 2021

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	I led the development of the idea in collaboration with Bruno Lacerda and Nick Hawes. I performed the literature review with assistance from Bruno Lacerda. I led the work on the mathematical formulation of the presented algorithm. The paper was written by me with support from Paul Duckworth, Bruno Lacerda, and Nick Hawes.		
Signature		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

B. ***Exploiting Horizon Posterior Distributions for Efficient Planning as Inference***

This paper introduces an Expectation-Maximization (EM) algorithm for indefinite-horizon Markov Decision Processes (MDPs), conceptualized through a planning as inference framework. Our methodology leverages a posterior distribution over the horizon length, enabling us to optimize policies with enhanced computational efficiency. This approach offers a novel perspective compared to traditional dynamic programming methods by explicitly incorporating horizon considerations into the planning process. The key contributions of this work are: 1. Development of a new EM algorithm for solving indefinite-horizon MDPs. 2. Utilization of the horizon posterior for computational efficiency, providing a systematic method to determine the necessary iterations for deriving an optimal policy.

Now we discuss the potential utility of utilizing probabilistic inference in this paper.

- **Novel Capabilities (✓):** In this paper we present a novel algorithm by explicitly modelling the horizon as a random variable and exploiting that for computational efficiency. This can be seen as an adaptive version of policy iteration and outperforms existing methods on a variety of tasks.
- **Coherence (✓):** Both policy search and learning the horizon posterior are framed using probabilistic inference. This allows us to tab into a variety of inference methods to learn the optimal horizon posterior, such as variational inference.
- **Joint Optimization (✓) :** Jointly performing policy search and estimating the horizon posterior result in improved computational efficiency. This is because the algorithm for policy search is truncated based on the learned value of the horizon.
- **Automated Solvers (✓):** The entire process is solved using an EM-algorithm. Standard solvers can also be used for extensions of this problem to continuous and partially observable settings.
- **Connection to Literature (×) :** This paper does not highlight connections between fields.

The paper starts on the following page.

Exploiting Horizon Posterior Distributions for Efficient Planning as Inference

Anonymous submission

Abstract

Planning under uncertainty problems are usually formulated as Markov Decision Processes (MDPs), for which dynamic programming approaches can be used to find an optimal policy. Alternatively, one can consider planning as a probabilistic inference problem. In this paper, we present an Expectation-Maximization (EM) algorithm for indefinite-horizon, goal-oriented MDPs. Our proposed planning as inference formulation enables explicit reasoning on the horizon of the problem (i.e. how many timesteps the optimal policy is likely to take to reach the goal). Specifically, we show how, by exploiting a posterior distribution which represents the belief over the horizon length, we can achieve significant computational gains. Furthermore, we investigate the relationship between the dynamic programming and planning as inference solution methods for MDPs, drawing connections between our EM algorithm and policy iteration.

Introduction

Planning under uncertainty is a key problem in the field of artificial intelligence. A planning problem is typically modelled using a Markov Decision Process (MDP). An MDP consists of 1) a set of world states, 2) a set of actions, 3) a transition model, and 4) an objective function to be optimized (e.g. maximizing rewards over a sequence of timesteps). To solve MDPs, one must compute a policy, which determines the action required at each decision point to optimize the objective. Optimal policies are typically obtained using dynamic programming algorithms (Kolobov 2012; Sutton, Barto et al. 1998; Puterman 2014).

Approaches based on dynamic programming are anchored in Bellman's equations (Bellman 1957), which reflect the recursive property of future returns. In dynamic programming a state refers to a *world state* (e.g. a grid cell in a grid world), and we reason about the value of a state using its immediate reward and the expected value of future (world) states. Planning as inference approaches (Toussaint and Storkey 2006; Baioumy et al. 2021a; Attias 2003) provide an alternative perspective to solving an MDP. Specifically, a state in an inference problem represents a *random variable* over all world states at a given timestep. Probabilistic inference is used to find the optimal policy by maximizing the likelihood of the model.

A limited number of works have explored the potential of probabilistic inference in planning. Most of this work assumes that the horizon length is finite and fixed a priori (Attias 2003; Verma and Rao 2005; Mukadam, Yan, and Boots 2016). This is limiting since many problems are best modelled using an MDP with an *indefinite horizon* (a finite but unknown horizon). This includes goal-oriented problems such as stochastic shortest path MDPs (SSP MDPs) (Kolobov 2012), which require computing a policy that minimizes the expected cumulative cost to reach a set of goal states; and probabilistic reachability problems (Steinmetz, Hoffmann, and Buffet 2016), which require computing a policy that maximizes the probability of reaching a set of goal states. In such settings the horizon length, i.e. the number of steps needed to reach the goal, is unknown a priori. Thus, choosing any fixed horizon can results in sub-optimal behaviour and inefficient policy search.

Planning as inference for MDPs without assuming a fixed horizon a priori has been addressed previously in (Toussaint and Storkey 2006), but only in the context of *infinite-horizon discounted* reward maximization. The approach is based on modeling the horizon itself as a random variable and performing inference on a mixture model.

In this paper we make two contributions. First, we present the first approach for solving indefinite-horizon MDPs using probabilistic inference. This modifies the approach in (Toussaint and Storkey 2006) to address the undiscounted indefinite-horizon case (rather than the discounted infinite-horizon case). Second, we present a novel approach for gaining significant computational efficiencies during planning by explicitly reasoning about the *horizon posterior*. In the goal-oriented setting, this refers to computing a distribution over how many timesteps an agent is likely to require to optimally reach the goal. For a given policy this distribution is influenced by a) the reward function, b) the initial state, and c) the structure of the world, e.g. the location of goal states. By computing the horizon posterior, we can leverage additional information (such as the support of the horizon posterior distribution) that allows us to terminate the optimization early. Intuitively, this provides a principled way of finding the maximum number of iterations needed to find an optimal policy.

This paper considers finding *exact solutions* for an MDP, and aims to connect probabilistic inference to policy iteration and its extensions. Our goal is to lay the foundation for using horizon posteriors in planning. An inherit limitation

of any exact method is scalability. Thus, in our experiments we consider discrete MDPs with relatively small state and actions spaces. In particular, we evaluate our approach on a set of domains from OpenAI gym (Brockman et al. 2016) showing that planning with horizon posteriors outperforms policy iteration in terms of the number of sweeps through the state space during policy computation. However, in Section we also provide a discussion on how the presented method can be extended for large and continuous state-spaces using approximate solution methods.

Related work

Many problems can be framed as probabilistic inference, including state-of-the-art approaches in reinforcement learning (Haarnoja et al. 2018; Levine 2018), localization and mapping (Dellaert and Kaess 2017), motion planning (Mukadam et al. 2018; Toussaint 2009), optimal control (Watson and Peters 2021) and fault-tolerant control (Baioumy et al. 2021b). This allows researchers to exploit the vast literature of probabilistic inference methods to find efficient solutions to the problem at hand (Murphy 2012; Koller and Friedman 2009).

In this paper, we focus on planning under uncertainty. Only a small number of planning approaches have been proposed that build on probabilistic inference. Early work in (Attias 2003) translated the problem of planning to a problem of inference. A Markovian model is assumed with a prior on the initial and the final states. Then a maximum a posteriori approach is used to find a plan (a sequence of actions) that maximizes the probability of reaching the goal. This, crucially, assumes the horizon to be fixed in advance and computes an open-loop plan rather than a policy. Work in (Verma and Rao 2005, 2006) presents a similar (open-loop) approach based on the maximal probable explanation.

In the active inference community (Friston et al. 2017), algorithms based on Bayesian inference have been presented for planning under uncertainty (Da Costa et al. 2020; Millidge et al. 2020). These approaches can be used to find an optimal policy, but still assume a finite horizon fixed a priori. Several approaches in the robotics community have also used probabilistic inference for motion planning and trajectory optimization e.g. (Mukadam, Yan, and Boots 2016; Mukadam et al. 2018; Toussaint 2009). Similar to the above, a prior is enforced on the initial and final states and the horizon is fixed a priori.

In (Toussaint and Storkey 2006) a probabilistic inference approach that does not assume a fixed horizon was presented. The authors present a message-passing algorithm for finding the optimal policy of an MDP. However, they only consider the case of an infinite horizon with discounted rewards. This is not suitable for many problems where undiscounted rewards need to be optimized, such as SSP MDPs (Kolobov 2012) and probabilistic reachability (Steinmetz, Hoffmann, and Buffet 2016). The approach in (Toussaint and Storkey 2006) is flexible and was modified to solve continuous and partially observable MDPs (Toussaint, Charlin, and Poupart 2008; Toussaint, Harmeling, and Storkey 2006). This approach has also proven useful in many real world planning problems, such as scalable multi-robot planning (Kumar, Zilberstein, and Toussaint 2015).

Preliminaries

Markov Decision Process (MDP)

An MDP is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : S \times A \times S \to [0, 1]$ is the transition function, $\mathcal{R} : S \times A \times S \to [0, 1]$ is the normalized reward function, and $\gamma \in (0, 1]$ is the discount factor. The expected reward of applying action a in state s is $\mathcal{R}(s, a) = \sum_{s' \in S} T(s, a, s') \cdot \mathcal{R}(s, a, s')$. A policy $\pi : S \to Dist(A)$ maps a state to a distribution over action choices, and we use the shorthand $\pi_{s,a} := \pi(s)(a)$ to denote the probability of choosing to take action a in state s under π. If π is *deterministic* we will write $\pi(s)$ to denote the action prescribed by π in state s.

For a policy π, let the random variables S_t be the state after t timesteps and A_t be the action selected by π at S_t. Given this, the expected value of π is $V^\pi = \mathbf{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(S_t, A_t)\right]$. The goal is to find the *optimal policy* π^* with the maximum expected reward, i.e., $\pi^*(s) = \operatorname{argmax}_\pi V^\pi$.

Solving an MDP can be achieved using standard dynamic programming algorithms such as value iteration or policy iteration. These approaches use the fact that there exists a deterministic optimal policy (Kolobov 2012). Policy iteration is of particular interest for this work. It is comprised of two steps, *policy evaluation* and *policy improvement*. In policy evaluation, for a given policy π, the state value function $V^\pi(s)$ is recursively evaluated until convergence: $V^\pi(s) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{R}(s, \pi(s), s') + \gamma V^\pi(s')\right]$. In the policy improvement step, an updated policy can then be computed from V^π as $\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a)$ for each state $s \in S$, where the state-action value function is defined as $Q(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V^\pi(s')]$. Iterating between these two steps guarantees convergence to an optimal policy (Kolobov 2012).

Probabilistic Reachability (MaxProb)

We will focus our experiments on *probabilistic reachability* problems, where the agent has a set of absorbing goal states $\mathcal{S}_g \subseteq \mathcal{S}$, and the aim is to find a policy π that maximizes the probability of reaching S_g. This setting is often referred to as MaxProb (Steinmetz, Hoffmann, and Buffet 2016), and is a canonical problem in probabilistic model-checking (Forejt et al. 2011). MaxProb can be solved by assigning a reward of 1 to the goal state (i.e. $\mathcal{R}(s, a, s_g) = 1$ for every $a \in \mathcal{A}$, $s_g \in \mathcal{S}_g$) and zero for all other states. Additionally, the discount factor is set $\gamma = 1$. This transforms MaxProb into a standard reward-maximization problem where policy iteration can be used. Furthermore, since goal states are assumed to be absorbing, the maximum reward attainable for any policy is 1.

Policy Search as Probabilistic Inference

An approach to compute an optimal policy for infinite-horizon discounted MDPs based on probabilistic inference is presented in (Toussaint and Storkey 2006). This approach

does not assume a fixed horizon a priori and works for general reward functions, rather than just probabilistic reachability. However, it is not valid for cases where $\gamma = 1$.

To represent an MDP without assuming a fixed horizon, (Toussaint and Storkey 2006) defines a *mixture of final reward MDPs*. A *final reward MDP* is an MDP of finite length T where a reward is emitted at the last timestep only. Recall that the random variables S_t and A_t denote the state after t timesteps and the action selected at S_t, respectively. Furthermore, let r be a random variable over the reward emitted at the last timestep. The (joint) probability of any path of length T occurring and reward r being emitted at time T can be factorized as:

$$P\left(r, S_{0:T}, A_{0:T} \mid T; \pi\right) = P\left(r \mid A_T, S_t\right) P\left(A_0 \mid S_0; \pi\right) \\ P\left(S_0\right) \cdot \prod_{t=1}^{T} P\left(A_t \mid S_t; \pi\right) P\left(S_t \mid A_{t-1}, S_{t-1}\right), \tag{1}$$

where $P(S_0)$ is the distribution over the initial state; $P(A_i|S_i;\pi)$ is the distribution over the action taken in the i-th step according to π, i.e. $P(A_i = a|S_i = s;\pi) = \pi_{s,a}$; and $P(S_t|S_{t-1}, A_{t-1})$ is defined according to the transition dynamics, i.e. $P(S_t = s'|S_{t-1} = s, A_{t-1} = a) = \mathcal{T}(s, a, s')$. The reward variable r is defined as a binary random variable $r \in \{0, 1\}$. Translating an arbitrary reward function to a reward distribution can be achieved by normalising the reward function $\mathcal{R}(s, a)$ between its minimum and maximum. In this case the probability of receiving a binary reward of 1 becomes equivalent to the scalar reward such that $P(r = 1|S_t = s, A_t = a) = \mathcal{R}(s, a)$. For more details see Appendix .

A *full MDP* emits a reward at every timestep and can be represented as a mixture model of final reward MDPs. This is represented by the joint distribution:

$$P\left(r, S_{0:T}, A_{0:T}, T; \pi\right) = P\left(r, S_{0:T}, A_{0:T} \mid T; \pi\right) P(T), \tag{2}$$

where $P(T)$ is a prior over the horizon, and is chosen to be a geometric distribution $P(T) = \gamma^T(1 - \gamma)$, where γ is the discount factor of the MDP. This leads to the horizon length being defined by a distribution, rather than a fixed value. In the case of the geometric distribution, this results in an *infinite* horizon.

To optimize rewards, we assume all binary reward variables have been observed to be 1. This indicates that for every timestep the agent obtained the maximum reward possible. We then aim to compute the values of the hidden variables that maximize the likelihood of this model. By doing so, we compute the policy that maximizes the likelihood of observing the maximum reward at each timestep. Additionally, as shown in (Toussaint and Storkey 2006), maximizing the likelihood of this model is equivalent to maximizing the value function:

$$L^{\pi}(s) = P(r = 1|S_0; \pi) = (1 - \gamma)V^{\pi}(s). \tag{3}$$

Thus, any inference algorithm that maximizes the likelihood of this model can be used to find the optimal policy of this model. Note that when $\gamma = 1$ the RHS of Equation 3 is zero, thus the approach in (Toussaint and Storkey 2006) is not able to maximize V^{π} in the undiscounted case.

Indefinite Horizon Planning as Probabilistic Inference

This section introduces the first contribution of the paper, adapting the method proposed in (Toussaint and Storkey 2006) to the case where $\gamma = 1$. We do so by changing the choice of the time horizon prior $P(T)$ from a geometric distribution to a uniform distribution.

Defining an indefinite-horizon MDP

We define an indefinite-horizon MDP in terms of a mixture of finite MDPs with only a final reward variable, similarly to Equation 2. However, the full MDP model for an indefinite horizon differs from Equation 2 in the choice of the horizon prior $P(T)$. The choice of $P(T)$ affects how the model handles discounting. As we have seen, choosing a geometric prior allows for solving the problem when $\gamma < 1$, but not when $\gamma = 1$.

When considering an indefinite-horizon MDP with undiscounted rewards (i.e. $\gamma = 1$), every reward counts equally in the objective. This can be framed in the probabilistic inference setting as a uniform prior over the horizon $P(T) = C$, where c is a constant. This constant is equal to $\frac{1}{t_{max}}$ given a horizon t_{max} corresponding to the maximum horizon for which it is possible to attain reward. However, t_{max} is not known a priori.

Fortunately, as will be shown in the next section, the likelihood of this model can be maximized without explicitly knowing the value of t_{max}. Since the prior is uniform, the posterior can be calculated exactly after normalization without knowing the value of t_{max} a priori.

Optimal policy search as probabilistic inference

The full MDP model in Equation 2 includes random variables over the states, actions, rewards and the horizon. Most of these are hidden (unobserved) random variables. Maximizing the likelihood of this model is indeed equivalent to maximizing the expected reward as:

$$L^{\pi}(s) = P(r = 1|S_0; \pi) = V^{\pi}(s). \tag{4}$$

Note how the only difference between this model and the model in (Toussaint and Storkey 2006) is the prior $P(T) = C$, which we choose as uniform rather than geometric. A key insight here is that the specific value of C is irrelevant for the optimization since our goal is to find a policy that *maximizes* the likelihood. Computing the exact value of the likelihood is not explicitly required; the likelihood of our model is exactly equivalent to the value function with normalized rewards. Exact connections between dynamic programming approaches and probabilistic inference are discussed in Appendix A.

Maximizing the likelihood of the model is achieved using an expectation maximization (EM) algorithm (Murphy 2012). Next, we discuss the an adaptation of the EM algorithm introduced by (Toussaint and Storkey 2006) which can be used to find the optimal policy π^* of our indefinite-horizon model. The E-step will, for a given π, compute a posterior over state-action sequences. The M-step then adapts the policy parameters π to optimize the expected likelihood with respect to the quantities calculated in the E-step.

E-step: a backwards pass in all finite MDPs. To simplify notation, we define $P(s' \mid s;\pi) := P\left(S_{t+1} = s' \mid S_t = s;\pi\right) = \sum_a \mathcal{T}(s,a,s')\pi_{s,a}$. To begin, we compute the first *backward message* $\beta_0(s) = P\left(r = 1 \mid S_T = s;\pi\right)$ for all $s \in \mathcal{S}$. This is referred to the as the 'base' and other quantities will be recursively computed from the base backwards in time. The base is computed as:

$$\begin{aligned}\beta_0(s) &= P\left(r = 1 \mid S_t = s;\pi\right) \\ &= \sum_a P\left(r = 1 \mid A_t = a, S_t = s\right)\pi_{as}.\end{aligned} \tag{5}$$

Conceptually, the quantity $\beta_0(s)$ is the expected reward if there is only one timestep remaining. Then, we recursively compute all the other backward messages. We use the index τ to indicate a backwards counter starting at T and decreasing to $t = 0$. This means that $\tau + t = T$, where T is total (unknown) horizon length. These messages are computed as:

$$\begin{aligned}\beta_\tau(s) &= P\left(r = 1 \mid S_\tau = s;\pi\right) \\ &= \sum_{s'} p(s' \mid s;\pi)\beta_{\tau-1}(s').\end{aligned} \tag{6}$$

Intuitively, the backward messages $\beta_\tau(s)$ are the expected rewards if one receives a reward at the last time step only. So, β_2, is the expected reward if the agent follows the policy π for two timesteps and only receives a reward exactly after two timesteps. Using these messages, we can compute a state-action value function dependent on τ as:

$$\begin{aligned}q_\tau(s,a) &= P(r = 1 \mid S_{T-\tau} = s, A_{T-\tau} = a, T = t + \tau;\pi) \\ &= \begin{cases} \sum_{s'} \mathcal{T}(s,a,s')\beta_{\tau-1}(s') & \tau > 1 \\ P\left(r = 1 \mid A_t = a, S_t = s\right) & \tau = 0. \end{cases}\end{aligned} \tag{7}$$

Marginalizing out time, we get the state-action value-function:

$$\begin{aligned}&P\left(r = 1 \mid A_T = a, S_t = s;\pi\right) \\ &= \frac{1}{C}\sum_\tau P(T)q_\tau(a,s) = \sum_\tau q_\tau(a,s),\end{aligned} \tag{8}$$

where C is a normalization constant equal to the prior of the horizon length. $P\left(r = 1 \mid S_{T-\tau} = s, A_{T-\tau} = a;\pi\right)$ is the probability of getting a maximum reward given a state and action, the following π. It is equivalent to the $Q(s,a)$ function computed in policy iteration. For more details on the connection to dynamic programming, see Appendix A.

M-step: the policy improvement step. The M-step then maximizes the log-likelihood $L^\pi(s) = P(r = 1|S_t = s, A_t = a;\pi)$ with respect to the new policy parameters π'. Given that the optimal policy for an MDP is deterministic, a greedy M-step can be used. This update is as follows:

$$\pi' = \operatorname*{argmax}_a P\left(r = 1 \mid S_t = s, A_t = a;\pi\right) \tag{9}$$

This update guarantees convergence to the optimal policy, since the optimal policy can be proved to be deterministic (Kolobov 2012). The M-step discussed here is exactly equivalent to a policy improvement step in policy iteration.

Exploiting the Horizon Posterior Distribution

This section discusses the second contribution of the paper. We present an approach for computing horizon posterior distributions and exploit them for computational efficiency gains. This leads to faster convergence to the optimal policy. Our key insight is to compute a posterior over the time horizon T required for optimal behaviour given a policy. Crucially, this approach works for any arbitrary initial state distribution.

In Section , we defined a probabilistic model which included only a prior $P(T)$ over the horizon parameter , then in the E-step we computed the probability of having maximum rewards given the states and actions, i.e. the posterior over rewards $P\left(r = 1 \mid S_t = s, A_t = a;\pi\right)$. Now, we will additionally compute the posterior over the time horizon and exploit it for computational gains.

Under the probabilistic inference approach for planning, the horizon length determines how many times we iterate through the state-space in the E-step until convergence (equivalent to a policy iteration step). We refer to one sweep through the state space as an epoch.

Given an initial state S_0, we can elicit meaningful information about the horizon of the current problem. The idea is that, at every iteration in the E-step, we compute the posterior distribution over the time horizon parameter $P(T|r = 1, S_0;\pi)$. This distribution assumes that we have obtained maximal rewards and depends on the initial state S_0. Then, the computed posterior can be used as an updated prior in the next iteration of the EM algorithm.

Computing the horizon posterior

Before showing how the knowledge of the horizon posterior can be exploited for computation efficiency, we present a way to exactly compute $P(T|r = 1, S_0 = s_0;\pi)$ using a message passing algorithm. We perform this by introducing 'forward messages' α. The β values presented in the previous section are considered 'backward messages', i.e. they are iteratively computed backwards over time (with index τ, which decreases after every iteration). The actions and states at a timestep are dependent on the future rewards, so backward messages β carry the relevant information from future rewards backwards to previous temporal states.

On the contrary, forward messages (α messages), carry information about which world states in $\mathcal{S}$ the agent will likely occupy at time step t given a policy and a known distribution over the initial state. One can assume any initial state distribution. The 'base' α message α_0 is then given as:

$$\alpha_0(s) = P(S_0 = s). \tag{10}$$

We then sequentially compute the other forward messages following policy π, with the iterative update rule:

$$\alpha_t(s) = P(S_t = s|S_0;\pi) = \sum_{s'} P(s|s';\pi)\alpha_{t-1}(s'). \tag{11}$$

The likelihood of receiving maximum rewards for any specific horizon T given that we start in the initial state s_0 and

follow a policy π can be computed as:

$$L_T^\pi(s) = P(r = 1|T = t + \tau, S_0; \pi) = \sum_s \alpha_t(s)\beta_\tau(s). \tag{12}$$

Then the horizon posterior can be computed as:

$$P(T|r = 1, S_0; \pi) = \frac{1}{C_2} L_T^\pi P(T) \tag{13}$$

where C_2 is a normalizing constant.

Crucially, this approach still yields computational efficiencies even for a uniform initial state distribution:

$$\alpha_0(s) = P(S_0 = s) = \frac{1}{|\mathcal{S}|} \tag{14}$$

for every $s \in \mathcal{S}$. Planning with horizon posteriors is thus a general method and can be used for arbitrary MDPs.

Computing support of the horizon posterior

Computing a posterior distribution over the time horizon parameter is interpreted as maintaining a belief about how likely the agent is to act optimally, given that it acts for T timesteps. From this, a straightforward way to gain computation efficiency is to compute the support of the distribution: the minimum and maximum value of non-zero probability. For most problems however, the support is unbounded, so we use a cut-off ϵ (chosen to be 10^{-8} in the experiments).

We can compute the support of the distribution and gain computational advantages by terminating the E-step early (since we know the maximum value for T). Pseudo code for the E-step while taking the cut-off horizon into account is provided in Algorithm 1. Intuitively, we are now maximizing the reward function $\sum_k r_k$ from T_{min} to T_{max}. This makes our algorithm equivalent to modified policy iteration (Puterman and Shin 1978) with a *learned value* for T_{max} adapting with every iteration.

Algorithm 1: E-step for Planning as Inference exploiting Horizon Posteriors

Input: MDP, π, $P(T)$
Output: $Q(s, a)$, $P(T)$
Initialize: β_0 (Eq. 5), α_0 (Eq. 10)
For t = 0 to T_{max} :
  Compute β_τ(s)
  Compute α_t(s)

Compute $q_\tau(s, a)$
Compute $P(T|r = 1, S_0; \pi)$
$T_{max} = \text{argmax}_H[P(T = H|r = 1, S_0) > \epsilon]$
Compute T_{min} (or set to 0)
$P(T) = Uniform(T_{min}, T_{max})$
$Q(s, a) = \sum P(T)q_\tau(a, s))$
Return: $P(T)$, $Q(s, a)$

Relationship to heuristic search

Planning with horizon posteriors exploits the domain structure (and optionally the knowledge of the initial state) for computational speed-ups. The idea of exploiting knowledge of the initial state is key in heuristic search algorithms such as LAO$*$ (Hansen and Zilberstein 2001) and LRTDP (Bonet and Geffner 2003). Since a large number of states are not reachable given an initial state s_0, it is unnecessary to compute a complete optimal policy, and a partial policy suffices, as long as that policy is defined for all reachable states. Heuristic search algorithms compute a partial policy given the initial state, and thus also gain computational speed-ups.

Planning with horizon posteriors is fundamentally different since it computes a *complete optimal policy* rather than a partial policy. As discussed, exploiting the horizon for computational gains can be achieved without knowledge of the initial state in the horizon posteriors case. The seed message α_0 would simply be uniform.

Planning with horizon posteriors can be seen as an orthogonal method to heuristic search. One could also use probabilistic inference to compute partial policies similar to work in heuristic search.

Although computing partial policies is out of the scope of this paper, we provide one such example here. First, we can compute

$$\kappa_{t\tau}(s) = P\left(S_t = s \mid r = 1, T; \pi\right) = \frac{1}{L_{t+\tau}^\pi} \beta_\tau(s)\alpha_t(s),$$

where the likelihood $L_{t+\tau}^\pi$ plays the role of a normalization constant. Then the posterior probability of visiting a certain state s within a rewarded trajectory is given as

$$P(s \in S \mid r = 1; \pi) = \sum_T \frac{P(T)L_T^\pi}{L^\pi} \left[1 - \prod_{t=0}^{T} \left[1 - \kappa_{t,T-t}(s)\right]\right]. \tag{15}$$

Given an initial state, there is a large part of the state space where $P(s \in S \mid r = 1; \pi) = 0$. This set can be pruned as part of the inference process to compute partial policies.

Other ways to exploit the horizon

One could also directly use the horizon posterior as the prior for the next iteration of the EM algorithm. To simplify this section we use the notation $H_k = P(T = k|r = 1, S_0 = s_0; \pi)$, e.g. $H_3 = P(T = 3|r = 1, S_0 = s_0; \pi)$. In this case, the reward being maximized will be discounted according to the probability distribution of the horizon: $\sum_k H_k r_k$. This means that location on the horizon where we expect to see a reward will count with a higher weight than the ones where we did not see a reward.

Maximizing the likelihood of this model would then *not* be equivalent to maximizing the original reward function (whether it is discounted or undiscounted). Further investigation of this approaches is left for future work, including a proof that this is guaranteed to converge to the optimal policy.

Another way in which the horizon could be exploited is to approximate it. We can approximate the distribution $p(T = H|r = 1, S_0 = s_0; \pi)$ with a variational distribution $q(T)$ such that $q(T) = \gamma(1 - \gamma)^T$ is a geometric distribution.

This is reduced to optimization problem where we learn the parameter γ that minimizes the KL-divergence between the two distributions (Fox and Roberts 2012) given as:

$$KL\left[q(T)||p(T|r,S_0;\pi))\right] = \sum_T q(T)\ln\frac{q(T)}{p(T|r,S_0;\pi)}$$
$$= -\sum_T q(T)\ln\frac{q(T)}{p(T,r,S_0;\pi)} + \ln p(r,S_0;\pi)$$
$$= \mathcal{F} + \ln p(r,S_0;\pi) \geq 0. \quad (16)$$

The parameter γ corresponds to a discount factor. This would 'adjust' the discount factor to fit the problem at hand. For instance, if we consider a problem where the initial state is very close to the reward-omitting states, the discount factor will remain low. While if the highest rewards are far away from the initial state, the discount factor will proportionally increase. Again, there is no proof that this approach would maximize the original reward function since the discount factor is dynamically changing. Analysis on whether such an approach would be optimal is left for future work. Brief results on using variational inference to compute the horizon posterior are provided in the appendix.

Results

Domains

To validate the approach presented in this paper, we use three environments from OpenAI gym (Brockman et al. 2016).

Frozen Lake: a grid world environment with one goal state. The environment is highly stochastic: actions move the agent to the intended state with a probability of $\frac{1}{3}$, and a probability of $\frac{1}{3}$ in each orthogonal direction. Trap states exist where the agent receives no reward. Both the goal and the trap states are absorbing states. Thus, this is an example of a MaxProb problem (Steinmetz, Hoffmann, and Buffet 2016): maximizing the rewards is equivalent to maximizing the probability of reaching the goal state.

Maze navigation: a grid world with one goal state. The environment is deterministic. Walls exist that the agent can not move through, however, the walls are not trap states.

Cliff World: a grid world where an agent incurs a cost of 1 for every transition. Additionally, next to the goal are 'cliff states', which incur a cost of 100 and move the agent back to the initial state. The goal state is the only absorbing state and incurs no cost. This is an example of a stochastic shortest path MDP (Kolobov 2012). We modify the standard Cliff World environment (from (Sutton, Barto et al. 1998)) to be stochastic rather than deterministic. An action has a probability of 0.9 of moving the agent to its intended state.

Planning with horizon posteriors given an initial state

To illustrate the properties of the horizon posterior, consider the 8x8 Frozen Lake environment in Figure 1 (Left). The environment is highly stochastic and there is only one reward to be received at the goal state (marked green). The agent starts with a random equiprobable policy π_{random} (uniform for all actions).

The distribution $P(T|r=1,S_0;\pi_{random})$, specifies the probability of getting a maximal reward within a certain number of timesteps, given that it follows the random policy. This quantity depends on the initial state. Figure 1 (Middle) shows this distribution for a sample of four potential initial states. It is clear that when the agent is close to the goal (e.g. $S_0 = D$), the agent is likely to reach the goal in a small number of timesteps compared to other initial states. Thus by exploiting horizon posteriors, we can terminate the EM algorithm after about 40 epochs (given that the horizon posterior is close to zero). This is not the case for the other initial states.

Now consider the case when $S_0 = A$. From Figure 1 it is clear that there are two paths from the state A to the goal state. A short path surrounded by absorbing trap states (colored red) and a longer path far from trap states. The horizon posterior for $S_0 = A$ is thus a multi-modal distribution with a mode representing the two possible paths (one shorter path through the middle of the trap states, and a second longer and safer path around the trap states). This can be seen as the blue curve in Figure 1 (Middle) and more clearly in Figure 1 (Right).

During the EM algorithm, we iteratively compute policy improvements, which leads to changes in the agent's horizon posterior. Figure 1 (Right) shows an initial random policy, an intermediate policy and the optimal policy. Note that the distribution is increasingly sharper around the second peak since the optimal policy takes the longer but safer path to the goal.

Computational speed-ups using the horizon posterior

In this section, we compare planning with horizon posteriors and standard policy iteration. In both settings *we do not assume any knowledge about the initial state*. In both settings we assume a cutoff of $\epsilon = 10^{-8}$. The value ϵ is used in policy iteration to terminate the policy evaluation step, and in inference it is used to calculate the support of the horizon posterior.

To allow for a fair comparison, we report the total number of epochs (sweeps through the state space) each method requires to converge to the optimal policy. For planning as inference this includes counting both β message as well as α messages.

Exploiting horizon posteriors consistently outperforms policy iteration in terms of number of epochs required for conversion for every environment across all our domains. A sample of the results is summarized in Table **??**. The computational gain for Maze Navigation is negligible for all sizes of the problem. This is due to it being a deterministic environment. However the gains for stochastic environments are shown to be substantial. Note that policy iteration and planning as inference are both deterministic methods, thus for the same environment the results have a standard deviation of zero and thus are not reported. The results in the above table are for a single environment per domain. For instance, Cliff World has a specific configuration of obstacles

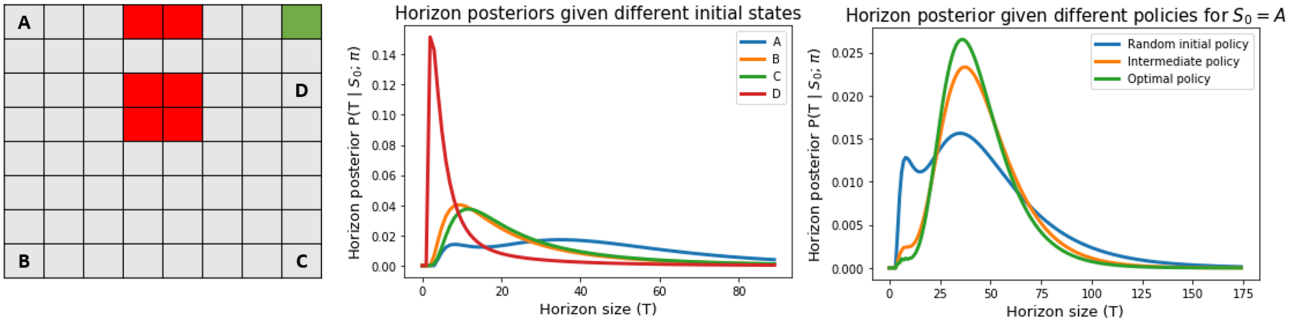


Figure 1: **(Left)** An example 8x8 grid world. The green states at the top right corners is the goal state, red states are trap states (absorbing states with no rewards) and finally A, B, C and D are different potential initial states. **(Middle)** The horizon posterior given different initial state and a random policy. **(Right)** The horizon posterior given the initial state A for different policies obtained during the EM algorithm. Note how the blue curve in both plots is the same.

Environment	Computational speed up (%)
Frozen Lake (64 states)	30.1
Frozen Lake (6400 states)	39.2
Maze Navigation (10000 states)	2.5
Stochastic Cliff world (24 states)	44.0
Stochastic Cliff world (4800 states)	42.8

and goal states. The computational gains for planning with horizon posteriors are however sensitive to the location of trap states. To investigate this further we generate a number of frozen lake environments. For every grid size we generate $50 \cdot |\mathcal{S}|$ environments and run both policy iteration and planning with horizon posteriors. We present results in Figure 2 for grid sizes varying between 2x2 and 14x14. For a grid size of 14x14, we generate $14 \cdot 14 \cdot 50 = 9800$ configurations, then take the mean and standard deviation across them. The mean represents the number of epochs it takes to converge per policy. The percentage of trap states in every grid world is set to 10%.

These experiments are conducted on a 2.6 GHz Intel Core i7 PC with 16 GB of RAM.

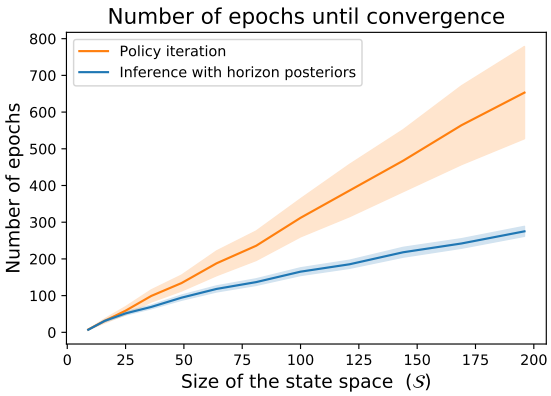


Figure 2: Comparison between standard policy iteration and planning as inference with horizon posteriors. For different sizes of the state space, $50 \cdot |\mathcal{S}|$ environments are generated with 10% trap states at random locations.

Discussion and conclusion

In this paper we frame planning with an *indefinite* horizon MDP as a probabilistic inference problem and present an Expectation-Maximization (EM) algorithm to compute the optimal policy. During the E-step of our algorithm, we explicitly compute a posterior distribution over the time horizon (horizon posterior) and exploit its support for computational efficiencies. Our results validate that exploiting the horizon posterior consistently results in computational efficiencies for MaxProb problems (Forejt et al. 2011; Steinmetz, Hoffmann, and Buffet 2016) and stochastic shortest path MDPs (Kolobov 2012). Crucially, the computational gains are still significant even without any knowledge of the initial state.

In this paper we focused on developing an exact solution method that exploits the horizon posteriors for computational speed-ups. However, like any other exact solution methods, this is only applicable to small and discrete state-spaces. Scaling this approach to large state-spaces can be achieved by utilizing approximate methods in the E-step of the EM algorithm. For instance, in (Vlassis et al. 2009) a Markov Chain Monte Carlo (MCMC) algorithm is used to scale the original methods of (Toussaint and Storkey 2006). Additionally, function approximations can be used to represent the value function and policy, as is in deep reinforcement learning methods (Sutton, Barto et al. 1998; Haarnoja et al. 2018). Finally, as described in Appendix C, our method can be used to compute partial policies as in heuristic search methods.

References

Attias, H. 2003. Planning by probabilistic inference. In *AISTATS*.

Baioumy, M.; Lacerda, B.; Duckworth, P.; and Hawes, N.

2021a. On Solving a Stochastic Shortest-Path Markov Decision Process as Probabilistic Inference. *arXiv preprint arXiv:2109.05866*.

Baioumy, M.; Pezzato, C.; Ferrari, R.; Corbato, C. H.; and Hawes, N. 2021b. Fault-tolerant Control of Robotic Systems with Sensory Faults using Decoupled Active Inference. In *European Control Conf.*

Bellman, R. 1957. A Markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *ICAPS*, volume 3, 12–21.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Da Costa, L.; Sajid, N.; Parr, T.; Friston, K.; and Smith, R. 2020. The relationship between dynamic programming and active inference: The discrete, finite-horizon case. *arXiv preprint arXiv:2009.08111*.

Dellaert, F.; and Kaess, M. 2017. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2): 1–139.

Forejt, V.; Kwiatkowska, M.; Norman, G.; and Parker, D. 2011. Automated verification techniques for probabilistic systems. In *International school on formal methods for the design of computer, communication and software systems*, 53–113. Springer.

Fox, C. W.; and Roberts, S. J. 2012. A tutorial on variational Bayesian inference. *Artificial intelligence review*, 38(2): 85–95.

Friston, K.; FitzGerald, T.; Rigoli, F.; Schwartenbeck, P.; and Pezzulo, G. 2017. Active inference: a process theory. *Neural computation*, 29(1): 1–49.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hansen, E. A.; and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2): 35–62.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Kolobov, A. 2012. *Planning with Markov Decision Processes: An AI Perspective*, volume 6. Morgan & Claypool Publishers.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2015. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research*, 53: 223–270.

Levine, S. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Millidge, B.; Tschantz, A.; Seth, A. K.; and Buckley, C. L. 2020. On the Relationship Between Active Inference and Control as Inference. In *International Workshop on Active Inference*, 3–11. Springer.

Mukadam, M.; Dong, J.; Yan, X.; Dellaert, F.; and Boots, B. 2018. Continuous-time Gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11): 1319–1340.

Mukadam, M.; Yan, X.; and Boots, B. 2016. Gaussian process motion planning. In *2016 IEEE international conference on robotics and automation (ICRA)*, 9–15. IEEE.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Puterman, M. L.; and Shin, M. C. 1978. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11): 1127–1137.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research*, 57: 229–271.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Toussaint, M. 2009. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, 1049–1056.

Toussaint, M.; Charlin, L.; and Poupart, P. 2008. Hierarchical POMDP Controller Optimization by Likelihood Maximization. In *UAI*, volume 24, 562–570.

Toussaint, M.; Harmeling, S.; and Storkey, A. 2006. Probabilistic inference for solving (PO) MDPs. *University of Edinburgh, School of Informatics Research Report EDI-INF-RR-0934*.

Toussaint, M.; and Storkey, A. 2006. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the 23rd international conference on Machine learning*, 945–952. ACM.

Verma, D.; and Rao, R. 2005. Graphical models for planning and imitation in uncertain environments. Technical report, Technical Report 2005-02-01, Department of CSE, University of Washington.

Verma, D.; and Rao, R. P. 2006. Goal-based imitation as probabilistic inference over graphical models. In *Advances in neural information processing systems*, 1393–1400.

Vlassis, N.; Toussaint, M.; Kontes, G.; and Piperidis, S. 2009. Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, 27(2): 123–130.

Watson, J.; and Peters, J. 2021. Advancing trajectory optimization with approximate inference: Exploration, covariance control and adaptive risk. *arXiv preprint arXiv:2103.06319*.

Connections between the two views

The messages β computed during backward propagation are exactly equal to the value functions for a single MDP of finite time. For undiscounted rewards, the full value function can therefore be written as the sum of the βs,

$$V_\pi(s) = \sum_T \beta_T(s)$$

since the prior over time $P(T)$ is a uniform prior. If $P(T)$ is not a uniform distribution, this would result in a mixture rather than a sum given as:

$$V_\pi(s) = \frac{1}{C_5} \sum_T P(T)\beta_T(s)$$

where C_5 is a normalizing constant. If $P(T)$ is a geometric distribution, then $C_5 = 1 - \gamma$. The value function would maximize discounted reward in that case.

The same applies to the relationship between to the Q-value function:

$$Q_\pi(a, s) = \sum_T q_T(a, s).$$

The E-step essentially performs policy evaluation which yields the classical value function. Given this relation to policy evaluation, the M-step performs an operation exactly equivalent to standard policy improvement. Thus, the EM-algorithm using exact inference is equivalent to Policy Iteration but computes the necessary quantities differently.

In the policy evaluation step, one iterates through the state space to update the value $V_\pi(s)$ for every policy until a termination criterion is met. Most commonly, an ϵ-greedy criterion is used. This means that we stop iterating though the state space once the maximum difference in $v_\pi(s)$ for any s is smaller than a constant ϵ.

Another well-known algorithm for computing an optimal policy is value iteration. Value iteration can be seen as policy iteration where we iterate through the state-space once, then perform a greedy policy update. This can be expressed as

$$V(s) \leftarrow \max_a \sum_{s'} P(s'|s, a)(\mathcal{R}(s, a, s') + \gamma V(s')). \quad (17)$$

Policy iteration typically converges faster since the policy typically converges before the value function does.

An alternative to either is *modified policy iteration* (Puterman and Shin 1978). Here we iterate through the state space T_{max} times then perform policy improvement. So rather than iterating until convergence, or just iterating once, we iterate a fixed number of time-steps. Notably, modified policy iteration still guarantees optimality but can converge much faster than either policy iteration of value iteration.

To summarize, modified policy iteration, one sweeps through the entire state space T_{max} times. Value iteration can be seen as a special case where $T_{max} = 1$. Standard policy iteration one sweeps through the state space to update the value $v_\pi(s)$ for every state until a termination criterion is met. An ϵ-greedy criterion is the most common, and means that we stop iterating though the state space once the maximum difference in $V_\pi(s)$ for any s is smaller than a positive small number ϵ.

The probabilistic inference algorithm presented in this paper is equivalent to modified policy iteration. Planning with horizon posteriors (algorithm 1) is similar to modified policy iteration, however, we learn T_max for every iteration (in the EM algorithm) rather than fix it initially.

Computing the reward distribution

The reward probability distribution $P(r_t|S_t, A_t)$ is defined differently than the reward function $\mathcal{R}$. Generally the reward function is define over any positive number, however, a distribution need to be normalized between 0 and 1, should integrate to 1. Thus the temporal reward variables r_t are defined as binary random variables $r_t \in \{0, 1\}$. Translating an arbitrary reward function to temporal rewards can be done by scaling the reward function $\mathcal{R}(s, a)$ (as defined in the previous section) between the minimum reward (min($\mathcal{R}$)) and maximum reward (max($\mathcal{R}$)) as:

$$P(r_t = 1 \mid A_t = a, S_t = s) = \frac{\mathcal{R}(a, s) - \min(\mathcal{R})}{\max(\mathcal{R}) - \min(\mathcal{R})}. \quad (18)$$

Any expression with $P(r_t = 1)$ can be though of as 'the probability of a reward being maximal' or 'the probability of the agent being optimal'. Thus, the probability of a reward being maximal given a state and an action is $P(r_t = 1 \mid A_t = a, S_t = s)$. Now we can reason about the highest possible reward for a state s and action a as one where $P(r_t = 1 \mid A_t = a, S_t = s) = 1$ and the lowest possible reward as $P(r_t = 1 \mid A_t = a, S_t = s)$ = 0. Any other reward will have a probability in-between, according to its magnitude.

Variational inference for computing the horizon posterior

As discussed in the paper, one could approximate the distribution $p(T = H|r = 1, S_0 = s_0; \pi)$ with a variational distribution $q(T)$ such that $q(T)$ is a geometric distribution $(1 - \gamma)^T\gamma$. This is reduced to optimization problem where we learn the parameter γ that minimizes the KL-divergence between the two distributions given by

$$\begin{aligned} KL\left[q(T)||p(T|r, S_0; \pi))\right] &= \sum_T q(T) \ln \frac{q(T)}{p(T|r, S_0; \pi)} \\ &= -\sum_T q(T) \ln \frac{q(T)}{p(T, r, S_0; \pi)} + \ln p(r, S_0; \pi) \\ &= \mathcal{F} + \ln p(r, S_0; \pi) \geq 0. \end{aligned} \quad (19)$$

The factor γ corresponds to a discount factor in the reward function. The effective horizon is then compute as $1/(1-\gamma)$. Note that there is no proof that this method would actually maximize the original reward function. Such analysis is left for future work.

For a sample grid world, this method was used to compute an approximate horizon posterior. After an initial random

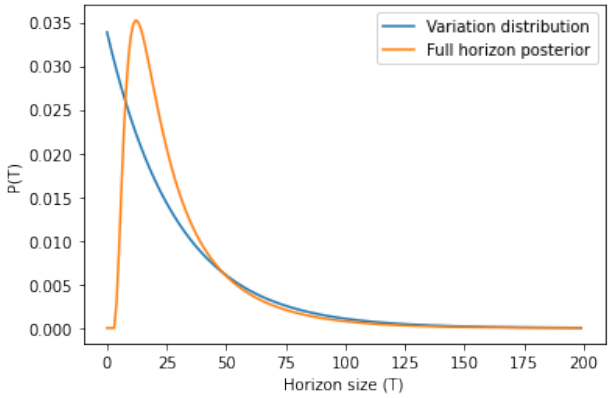


Figure 3: The horizon posterior of a sample grid world and the approximate horizon posterior.

policy, we can compute the approximate horizon posterior visualized in Figure 3.

The learned γ was 0.966, and thus the effective horizon is computed to be 29.4. As can be seen in the figure, this is slightly past the peak of the distribution and thus could be a reasonable cut-off.

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

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

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

Title of Paper	Exploiting Time-Horizon Posterior Distributions for Planning under Uncertainty.
Publication Status	Submitted for Publication
Publication Details	Mohamed Baioumy, Bruno Lacerda, Paul Duckworth and Nick Hawes, "Exploiting Time-Horizon Posterior Distributions for Planning under Uncertainty", Under review.

Student Confirmation

Student Name:	Mohamed Baioumy		
Contribution to the Paper	I led the development of the idea in collaboration with Nick Hawes. I performed the literature review with support from Bruno Lacerda. I led the work on the mathematical formulation of the presented algorithm. I carried out all the experiments. The paper was written by me with support from Paul Duckworth, Bruno Lacerda, and Nick Hawes.		
Signature		Date	30th of November 2023

Supervisor Confirmation

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

Supervisor name and title: Nick Hawes, Professor of AI & Robotics		
I confirm the correctness of the student's statement.		
Signature 	Date	13/12/2023

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

C. *Goal-based planning as inference*

This is a short section on goal-based planning. The aim of this section is to highlight the power of probabilistic inference to naturally exploiting the structure of the problem at hand. We additionally highlight the simplicity of extending this approach to continuous problems using automated toolboxes.

In goal-based planning problems, the agent has a set of absorbing goal states $s_g \subseteq \mathcal{S}$, and the aim is to find a policy π that maximizes the probability of reaching s_g. This setting is often referred to as MaxProb [34], and is a crucial problem in probabilistic model-checking [35]. This problem can be solved by assigning a reward of 1 to the goal state $\mathcal{R}(s, a, s_g) = 1$ for every $a \in \mathcal{A}$ and zero for all other states. Additionally, the discount factor is set to unity, $\gamma = 1$. This transforms MaxProb into a standard reward-maximization problem where policy iteration can be used. Furthermore, the goal state s_g is turned into an absorbing state, and thus the maximum reward attainable for any policy is 1.

This can be viewed a special case of an MDP, and specialized algorithms can be used to solved MaxProb. For example standard dynamic programming algorithms like policy iteration or value iteration can be adjusted to for MaxProb. Rather than initializing the value function randomly (or as 0 for every state), we initialize the goal states with a value of 1, we initialize the lost states (zero reward absorbing states) states with a value of 0. Neither is then updated after that. For all other states, we use a Bellman back-up without rewards:

$$V_\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, \pi(s), s'\right) \left[V_\pi\left(s'\right)\right].$$

This 'reward-less' Bellman back-up it identical to a regular Bellman back-up if you set all rewards to zero. In a nutshell, we initialize the goal state with a value of 1, then this value will be propagated to other states without the need to consider a reward function. This will be faster than standard policy iteration since in every single update there is less arithmetic to be performed. We are essentially ignoring all zero's in summation.

Now consider a full MDP model defined as:

$$P\left(r_{0:T}, s_{0:T}, a_{0:T} \mid T; \boldsymbol{\pi}\right) = P\left(a_0 \mid s_0; \boldsymbol{\pi}\right) P\left(s_0\right)$$
$$\cdot \prod_{t=1}^{T} P\left(a_t \mid S_t; \boldsymbol{\pi}\right) P\left(S_t \mid a_{t-1}, s_{t-1}\right) P(r_t | a_t, S_t).$$

This is the same model used the previous section. As shown, maximizing the likelihood for that model is equivalent to maximizing the reward. The algorithm discussed in the last section is exactly equivalent to Policy iteration.

For MaxProb cases, we intuitively know that there is only a reward at the very last time step. Thus, we can run the exact same message passing algorithm (which again can be automatically generated using existing toolboxes), but excludes all rewards besides the last. We are thus modifying *the model* to encode the knowledge of goal-based planning problems: you can only get any reward/value at the very last time-step. An algorithm would emerge where the seed $\beta_0(s)$ is still dependant on the rewards as: $\beta_0(s) = P\left(r = 1 \mid S_t = s; \boldsymbol{\pi}\right) = \sum_a P\left(r = 1 \mid A_t = a, S_t = s\right) \pi_{as}$.

However, all other backwards messages would have an update independent of rewards as:

$$\begin{aligned}
\beta_T(s) &= p\left(r = 1 \mid S_t = s; \pi\right) \\
&= \sum_{S_{t+1}} p\left(r, S_{t+1} \mid S_t; \pi\right) \\
&= \sum_{S_{t+1}} p\left(r \mid S_{t+1}, S_t\right) p\left(S_{t+1} \mid S_t\right) \\
&= \sum_{S_{t+1}} p\left(r \mid S_{t+1}\right) p\left(S_{t+1} \mid S_t\right) \\
&= \sum_{S_{t+1}} \beta_{t+1}(s) p\left(S_{t+1} \mid S_t\right)
\end{aligned} \tag{1}$$

Computing the seed messages, $\beta_0(i)$ is equivalent to initializing the value function of the goal state to 1, while the other backward messages are exactly equivalent to reward-less bellman backups done in MaxProb. We thus modified the model in a straightforward way (removed all but the last rewards), and the MaxProb emerged naturally by applying the exact same message passing algorithm.

It is already feasible to generate a message passing algorithm to solve the MDP. Now when one adds a modified model into the same solver, the emerging algorithm is equivalent to MaxProb. From the perspective of the researcher this moves the burden from devising algorithm and optimizing them to focusing on modelling the problem. In this case, it is arguably very intuitive to modify the model compared to the optimizer.

To evaluate this in practise, we consider a set of highly stochastic grid world environments (FrozenLake from OpenAI gym). The size of the grid world is 2^g, where g is real number between 2 and 12. For every environment size, we generate 10000 environments with the same percentage of lost states (zero reward absorbing states) (absorbing states with no rewards). There is one goal state. The process terminates when the agent finds a complete optimal policy.

Both inference take exactly the same number of steps in all simulations and converge to the same Q-function and policy. The version with final rewards only is on average 12% faster. In the case of the environment of size 2^{12}, it is 13.2% faster.

Consider now if one wishes to extend this problem to a case with continuous variables. Again, automated solvers (such as ForneyLab [33]) for probabilistic inference can greatly simplify this process. If one is working with discrete variable, and is using a categorical distribution, it would be defined for example as:

```
using ForneyLab

# Build the generative model
g = FactorGraph()

b = [0.7, 0.3]
@RV m ~ Categorical(b) # Prior
```

This shows how a prior can be added to a model. Once the rest of the model is specified, for instance the transition dynamics, a solver can be used. In ForneyLab, many message passing algorithms can be used with one line of code.

```
algo = messagePassingAlgorithm(m) # Build the algorithm code
```

Now consider the case of continuous random variables. The only change needed would be to define the random variable with a continuous distribution (e.g. a Gaussian) as its prior. This is very easily done by modifying the above code in the following way:

```
using ForneyLab

# Build the generative model
g = FactorGraph()

mean = 1.0
var = 2.0
@RV m ~ Gaussian(mean, var) # Prior

... more code ...

algo = messagePassingAlgorithm(m) # Build the algorithm code
```

This can be used to extend a solver for a discrete MDP to a continuous in a very easy manner. Since in both the discrete and the continuous case the model has the same structure, we simply just change a few lines of code.

In summary, this section highlights the benefits of coherence and automated solvers in the context of planning problems. We show that given it is possible to represent an MDP as a probabilistic inference problem, effective solutions can be found for special cases. In the case of goal-based planning, simply removing all non-final reward result in a computational again and algorithm equivalent to MaxProb. When extending this to more complex cases like continuous MDPs, only a few lines of code are needed to generate a solution to the new cases.

D. ***Monte Carlo Tree Search with Boltzmann Exploration***

In this short section, we aim to highlight two points, 1) many approaches in reinforcement learning are deeply connected to probabilistic inference, and 2) sampling-based methods vs parametric methods.

The paper [36] introduces two new algorithms for Monte Carlo Tree Search (MCTS) with Boltzmann Exploration: Boltzmann Tree Search (BTS) and Decaying ENtropy Tree-Search (DENTS). These algorithms address limitations in existing MCTS methods like the Upper Confidence Bound applied to Trees (UCT) and Maximum ENtropy Tree-Search (MENTS) by optimizing for reward maximization and maintaining the benefits of Boltzmann policies. The paper demonstrates the misalignment between the maximum entropy objective used in MENTS and reward maximization, hindering the convergence to optimal policies. BTS and DENTS are shown to be consistent in converging to reward-maximizing policies while retaining the exploratory advantages of Boltzmann policies. The effectiveness of these algorithms is validated through empirical analysis in various benchmark domains, including the game of Go.

This paper is not written from the perspective of probabilistic inference, and thus it is not included in this thesis. However, Maximum Entropy methods and Boltzman exploration can be directly liked to a probabilistic inference. Work in [18] shows the exact connections between these two fields. Maximum entropy methods are essentially a special class for models where variational inference is utilized. While it is not necessary to have the a background in variational inference to understand the resulting algorithm, a deep understand on such methods can aid researchers to discover new algorithms.

Second, the above paper utilizes a sampling-based method. This is similar to Markov Chain Monte Carlo method (e.g. [37]) when applied to the probabilistic inference model. In all planning paper present so far, exact infernce has been performed. Our work in [36] presented a contrast to that. Finally, in all work in the thesis, beliefs have been represented using Gasussians or Categorical variables. This is in contrast to particle based methods, such as particle filters [2], [3]. There thus two places where sampling can be used in planning: representing beliefs and sampling from estimate the target distribution.

VI. Discussion and Conclusions

In this thesis we investigate the utility of probabilistic inference as a general framework for decision-making in robotics. We tackle problems in several sub-fields in robotics: state-estimation, adaptive control, fault-detection, fault-tolerant control and planning under uncertainty. Traditional solutions in these fields are widely different. However, we approach all of them as probabilistic inference problems and standard methods in probabilistic inference are used to solve these problems. This is evidence that probabilistic inference is viable as a general framework for decision-making.

Second, with every contribution in the thesis, we assess the utility of solving the problem at hand as probabilistic inference. Essentially, even if it is possible to solve a planning problem or a fault-tolerant control problem as inference, is it *useful* to do so?

Despite the apparent successes, the application of probabilistic inference in robotics is not without its challenges. A significant limitation is the computational demand of these methods, particularly in real-time scenarios where rapid decision-making is critical. For instance, while variational inference techniques offer a robust framework for handling uncertainty, they can be computationally intensive, making them less suitable for low-power or time-critical applications. Furthermore, the sensitivity to model assumptions and the risk of overfitting are concerns that need careful handling. Additionally, only three papers contain real-world robot experiments. Finally, due to the generality of the thesis questions, when benchmarking against existing methods some nuances might have been lost. This thesis focuses on the unifying process of using PGMs to model domain-specific problems and then solving them using inference.

Throughout the thesis, we discuss five potential dimensions for utility: novel capabilities, coherence, joint optimization, automated solvers and connections between literature.

A novel capability refers to an advancement in a sub-field. For instance, in Chapter III, we present control strategies that outperform state-of-the-art Model Reference Adaptive Control (MRAC). This is valuable as it pushes the state-of-the-art. Not every paper in this thesis directly presents a novel capability. For instance, our work in Section IV-A presents a novel fault-tolerant control scheme based on active

inference. This method has merits as it is simple to implement and is computationally efficient. However, it is very sensitive to false positives every time a robot moves to a new goal state. This makes the approach not viable in practice. This paper shows some interesting properties for computational efficiency, but since the strategy is not viable in practice, it does not provide novel capabilities useful in FTC. In Section IV-B however, we combine the insights from this paper with the unbiased active inference controllers (u-AIC) and build a controller that is both robust and computationally efficient. This does provide novel capabilities and helps push the state-of-the-art in the sub-field of FTC.

Coherence refers to the property of modelling all aspects of the problem using one coherent framework. Essentially, the entire stack end-to-end is using probabilistic inference. The main benefit of coherence is simplicity. It can also help the researcher find valuable insights. First, simplicity is shown most clearly in the contrast between Section III-A and IV-C. In both papers, we perform control, state-estimation and estimate the co-variance of a distribution. In Section III-A, we don't model the co-variance as a random variable and instead manually account for complexities in our optimization. For instance, the co-variance is a strictly positive value, and a co-variance matrix is positive semi-definite. However, in Section IV-C we model the whole problem as probabilistic inference and such details are taken care of behind the scenes. This means we can use one solver for the whole stack. Mathematical formulation and notation are simplified. This additionally leads us to make valuable connections and discover new properties (the beta residual). Furthermore, in chapter V, modelling planning problems as probabilistic inference leads to several benefits. We identify the relationship between the horizon of the problems and convergence properties. We show how intuitive it is to adjust a standard MDP solver to optimize it for Goal-based planning. Such insights are hard to see from the dynamic programming perspective, but straightforward from the probabilistic inference perspective.

Joint optimization refers to jointly optimizing multiple problems instead of optimizing each separately. The main benefit of joint optimization is computational efficiency. This is most evident in Chapter IV. All three papers presented show how there are many redundant computations between problems, which can be exploited. Additionally, Section III-A presents an interesting algorithm where joint optimization is necessary for improved performance, not just computational efficiency. We

show in Section III-A that combining state-estimation and learning with control directly helps tune the gains of the controller.

Note that joint optimization is possible regardless of whether coherence is satisfied or not. In Section IV-A, Fault-detection and isolation (FDI) is not modelled as an inference problem, yet joint optimization is performed and computational gains are realized. In Section IV-C on the otherhand, FDI is modelled as inference and again joint optimization yields computational gains.

Automated solvers mean that one can use standard methods in probabilistic inference, such as probabilistic graphical model solvers to address the problem at hand. Instead of requiring researchers to devise a custom solution method at hand, it is sufficient to model the problem as an inference problem and leave the optimization to a standard solver. This is shown throughout the whole thesis. Rarely was there a need to devise a custom optimization method. Section V-C shows how an automated solver can be used to adapt an MDP solver for goal-based planning. The result is equivalent to well-known MaxProb algorithms.

Finally, connections between literature can be found when studying different field from the same lens. This can allow researchers from less mature fields to leverage a wealth of knowledge from other and aid interdisciplinary advancement. In this thesis this is demonstrated as many neuroscience-inspired methods have helped develop control strategies.

In conclusion, this thesis provides evidence that probabilistic inference can be used as a general framework for decision-making in robotics. For every contribution, we highlight the exact utility of using probabilistic inference.

A. General improvements

The general idea of this thesis was inspired by work in neuroscience. The free-energy principle and active inference are mentioned many times throughout the thesis. However, in hindsight, this limited the potential of the work in this thesis. At the start, I mainly looked at work related to active inference, rather than exploring more broadly how probabilistic graphical models are used in decision-making. Methods for planning and SLAM [17], [31], [32] could have provided more relevant pointers for directions to pursue in this thesis as they were more practical and directly applied

to robotics. For future researchers interested in this area, less emphasis should be put on work published in the active inference community.

Another point for general improvement of this work is putting more focus on practicality. Many papers only compared against one method. Although this is deemed enough for the paper, it generally limits the insight that could be gained from studying this work. Additionally, not all papers included experiments with real robots.

Finally, this thesis follows a model-based machine-learning approach. Learning models can be used within this framework. For instance, we approximate a function with a neural network within this paradigm. However, there is limited work in this thesis using neural networks. Gaussian Processes have been used for approximation, but no neural networks have been used.

B. Future work

As we look to the future, expanding the repertoire of probabilistic inference techniques in robotics holds promising potential. Beyond the narrow class of variational inference methods currently employed, exploring advanced probabilistic models like Gaussian processes and Bayesian nonparametrics could offer new insights, particularly in handling complex, high-dimensional data environments typical in robotics. These methods could enhance the robustness and adaptability of robotic systems in unpredictable settings, such as outdoor navigation or interaction with humans.

Future work can consider many avenues. Every paper in this thesis already highlights the potential for future work. However, on a higher level, a few more areas are promising for future work. First, one might argue that most benefits are a result of unifying approaches, not specifically probabilistic inference. If instead of probabilistic inference, one models all problems as constraint-optimization, would properties like coherence and joint-optimization arise and show the same benefits?

Second, we only scratched the surface in terms of probabilistic inference methods used. We used a narrow class of variational inference methods in most papers. Methods like variational message passing and Markov Chain Monte Carlo sampling have not been used.

Third, we have only considered three sub-fields in robotics. There is a wealth of other areas where the potential of probabilistic inference has not been investigated.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[2] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE proceedings F (radar and signal processing)*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[3] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," 2005.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[5] A. Kolobov, "Planning with markov decision processes: An ai perspective," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–210, 2012.

[6] K. J. Åström and T. Hägglund, *PID controllers: theory, design, and tuning*. Instrument society of America Research Triangle Park, NC, 1995, vol. 2.

[7] K. J. Åström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[8] G. Paviglianiti, F. Pierri, F. Caccavale, and M. Mattei, "Robust fault detection and isolation for proprioceptive sensors of robot manipulators," *Mechatronics*, vol. 20(1), pp. 162–170, 2010.

[9] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[10] E. Todorov, "General duality between optimal control and estimation," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4286–4292.

[11] K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the laplace approximation," *Neuroimage*, vol. 34, no. 1, pp. 220–234, 2007.

[12] K. Friston, "Hierarchical models in the brain," *PLoS computational biology*, vol. 4, no. 11, p. e1000211, 2008.

[13] K. J. Friston, J. Daunizeau, J. Kilner, and S. J. Kiebel, "Action and behavior: a free-energy formulation," *Biological cybernetics*, vol. 102, no. 3, pp. 227–260, 2010.

[14] K. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, pp. 127 EP –, 01 2010. [Online]. Available: https://doi.org/10.1038/nrn2787

[15] K. J. Friston, R. Rosch, T. Parr, C. Price, and H. Bowman, "Deep temporal models and active inference," *Neuroscience & Biobehavioral Reviews*, vol. 90, pp. 486–501, 2018.

[16] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[17] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Steap: simultaneous trajectory estimation and planning," *Autonomous Robots*, vol. 43, no. 2, pp. 415–434, 2019.

[18] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[19] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[20] H. W. Sorenson, *Kalman filtering: theory and application*. IEEE, 1985.

[21] N. T. Nguyen and N. T. Nguyen, *Model-reference adaptive control*. Springer, 2018.

[22] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2008.

[23] K. B. Pathak and D. M. Adhyaru, "Survey of model reference adaptive control," in *2012 Nirma University International Conference on Engineering (NUiCONE)*. IEEE, 2012, pp. 1–6.

[24] C. Pezzato, R. M. Ferrari, and C. Hernandez, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, 2020.

[25] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE Trans. on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[26] W. Hongm, C. Tian-You, D. Jin-Liang, and M. Brown, "Data driven fault diagnosis and fault tolerant control: some advances and possible new directions," *Acta Automatica Sinica*, vol. 35, no. 6, pp. 739–747, 2009.

[27] X. Yang, J. Gao, L. Li, H. Luo, S. X. Ding, and K. Peng, "Data-driven design of fault-tolerant control systems based on recursive stable image representation," *Automatica*, vol. 122, p. 109246, 2020.

[28] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[29] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 945–952.

[30] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[31] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 9–15.

[32] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[33] M. Cox, T. van de Laar, and B. de Vries, "A factor graph approach to automated design of bayesian signal processing algorithms," *International Journal of Approximate Reasoning*, vol. 104, pp. 185–204, 2019.

[34] M. Steinmetz, J. Hoffmann, and O. Buffet, "Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art," *Journal of Artificial Intelligence Research*, vol. 57, pp. 229–271, 2016.

[35] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, "Automated verification techniques for probabilistic systems," in *International school on formal methods for the design of computer, communication and software systems*. Springer, 2011, pp. 53–113.

[36] M. Painter, M. Baioumy, N. Hawes, and B. Lacerda, "Monte carlo tree search with boltzmann exploration," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[37] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, 2009.