Learning Search Strategies from Human Demonstrations

Dissertation (2016)

Submitted to the School of Engineering, Doctoral Program on Manufacturing Systems and Robotics

École Polytechnique Fédérale de Lausanne (EPFL)

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

by

Guillaume de Chambrier

Thesis Committee:

Prof. Dillenbourg Pierre, president of the jury
Prof. Aude Billard, thesis advisor
Prof. Pedro U. Lima, examiner
Dr. Thrish Nanayakkara, examiner
Dr. Mathew Magimai Doss, examiner

Lausanne, Switzerland
August, 2016

Abstract

Decision making and planning with partial state information is a problem faced by all forms of intelligent entities. The formulation of a problem under partial state information leads to an exorbitant set of choices with associated probabilistic outcomes making its resolution difficult when using traditional planning methods. Human beings have acquired the ability of acting under uncertainty through education and self-learning. Transferring our know-how to artificial agents and robots will make it faster for them to learn and even improve upon us in tasks in which incomplete knowledge is available, which is the objective of this thesis.

We model how humans reason with respect to their beliefs and transfer this knowledge in the form of a parameterised policy, following a Programming by Demonstration framework, to a robot apprentice for two spatial navigation tasks: the first task consists of localising a wooden block on a table and for the second task a power socket must be found and connected. In both tasks the human teacher and robot apprentice only rely on haptic and tactile information. We model the human and robot's beliefs by a probability density function which we update through recursive Bayesian state space estimation. To model the reasoning processes of human subjects performing the search tasks we learn a generative joint distribution over beliefs and actions (end-effector velocities) which were recorded during the executions of the task. For the first search task the direct mapping from belief to actions is learned whilst for the second task we incorporate a cost function used to adapt the policy parameters in a Reinforcement Learning framework and show a considerable improvement over solely learning the behaviour with respect to the distance taken to accomplish the task.

Both search tasks above can be considered as active localisation as the uncertainty originates only from the position of the agent in the world. We consider searches in which both the position of the robot and features of the environment are uncertain. Given the unstructured nature of the belief a histogram parametrisation of the joint distribution of the robots position and features is necessary. However, naively doing so becomes quickly intractable as the space and time complexity is exponential. We demonstrate that by only parametrising the marginals and by memorising the parameters of the measurement likelihood functions we can recover the exact same solution as the naive parametrisations at a cost which is linear in space and time complexity.

Keywords: Programming by Demonstration, POMDP, Reinforcement Learning, State Space Estimation (SSE)

Résumé

Raisonner et prendre des décisions afin de résoudre des problèmes avec une information partielle est une difficulté à laquelle doit faire face tout être intelligent. Les tentatives de résolution de problèmes spatiale dont l'information est partielle débouchent sur un nombre exorbitant d'actions possibles ayant chacune une probabilité de réussite propre. Ceci rend la résolution de tels problèmes difficile lors de l'emploi des méthodes de planning traditionnelles. L'objectif de ce mémoire est de créer des modèles mathématiques correspondant au raisonnement humain à l'égard de l'incertitude présente durant des tâches d'exploration dans le domaine de la navigation dans l'espace et de transférer ces modèles de raisonnement à un robot.

Nous modélisons les raisonnements cognitifs de localisation de sujets humains à l'aide d'une fonction de contrôle paramétrique dans un cadre de "Programmation par Démonstration". Cette modélisation est ensuite transférée à un robot apprenti afin que celui-ci réalise deux tâches de localisations. La première tâche consiste à localiser un bloc de bois posé sur une table. La seconde consiste à localiser une prise électrique, puis de la brancher à une prise murale. Durant l'accomplissement de ces deux tâches, l'humain et le robot apprenti ne disposent que d'informations haptique et tactile. Nous représentons les pensées cognitives de localisation de l'humain et du robot par une fonction probabilistique, mise à jour par un processus bayésien. Les résonnements humains sont modélisés par une distribution générative conjointe des pensées et actions (differential de l'effecteur) qui ont été enregistrées durant l'exécution des deux tâches. La première tâche a permis d'établir la relation entre pensée et action par une distribution conjointe, quant à la deuxième nous incorporant une fonction objective utilisée afin d'adapter les paramètres de la fonction de contrôle dans un cadre de renforcement d'apprentissage qui résulte en une amélioration considérable en matière de la distance parcourue pour l'accomplissement de la tâche.

Les deux tâches d'exploration mentionnées ci-dessus peuvent être considérées comme des problèmes de localisation-actif où l'incertitude est uniquement présente dans la relation entre la position de l'humain vis-à-vis du cadre de référence, le monde. Nous considérons maintenant un problème d'exploration où l'incertitude se trouve à la fois dans la position du robot (ou l'humain) et dans des aspects de l'environnement comme la position d'objets. Étant donné la nature non structurée de l'incertitude, un histogramme est choisi pour paramétrer la distribution conjointe des positions du robot et de l'environnement. Cependant, cette stratégie devient rapidement intenable; le coût de résolution devenant exponentiel en fonction du grand nombre de paramètres. Nous démontrons

qu'en appliquant les probabilités marginales aux paramètres des mesures, nous pouvons reproduire la solution identique de l'histogramme avec une complexité linéaire au lieu d'exponentielle.

Mots-clés: Programmation par démonstration, POMDP, Reinforcement Learning, Modèle espace d'états

À mon épouse bien aimée, amie et complice Jing
pour son amour inconditionnel

À mes parents et famille
pour leur soutien continu

Acknowledgments

I am grateful to my advisor Prof. Aude Billard for giving me the opportunity of undertaking a PhD under her supervision. I appreciate her oversight and valuable lessons regarding research and scientific questions. I am thankful for the inspiring discussions we had and the time she spent reading, correcting and providing valuable feedback to my papers.

I would like to acknowledge all my thesis examiners, Prof. Pedro U. Lima (IST), Dr. Thrish Nanayakkara (KCL) and Dr. Mathew Magimai Doss (IDIAP) for taking time to provide valuable comments and suggestions (including latex commands) on the earlier version of this manuscript. Many thanks also to Prof. Dillenbourg Pierre (EPFL) who served as president of my thesis committee.

I am thankful for the technical advice that Prof. Pedro U. Lima gave me through our frequent email exchanges during my second year. I am very grateful to Dr. Dimitri Ognibene with whom I have had many scientific discussions, for his advice and moral support which was of great help throughout my thesis; without his support I would have struggled far more. I would also like to extend a special thanks to Dr. J. Michael Herrmann who supervised me during my Master's thesis and provided great advice regarding family and research.

During my time at LASA I have met a wide spectrum of personalities which I wish to acknowledge here. First and foremost my office mates: Klas Kronander and Ajay Tanwani who left to reach his full potential (do you still jump up and down in front of the blackboard ?). Thank you Klas for your supportive words throughout our time in the lab, they always managed to motivate me when I as in doubt. Ajay, I am grateful of the time we spent together in our office, I know you are doing great and keep it up!

One of the very first person I met in the lab was Ashwini Shukla, who was tasked to demonstrate to me the capabilities of the iCube's skin. I am grateful for the laughs and good moments we spend together such as our trips with Miao Li to the swimming pool (which were mainly for gossiping and not swimming). My thanks extend to Miao Li, I enjoyed the way you think and view the world although I hope your geographic knowledge improves.

As for the old guard. I appreciate the discussion and gossip talk I had with Sahar El-Khour. I use to explain my research to you and you were helpful and supportive. Basilio Noris, thank you for passing onto me your pride and joy

"MLDemos". You have no idea how much fun I had with it during my teaching duties. I thank you also for helping me debug code during my first year, you remain to me a legendary coder. Seyed Mohammad, during my time in the lab you always had a smile on your face and I never saw you in a bad mood which was very uplifting. You gave my good advice regarding an "effect" to be wary of. I commend you sir for your wisdom. Lucia Ureche, you motivate me and some of the lads to go to the gym. I specially enjoyed our indoor cycling sessions and later our midday salad expeditions. Nicolas Sommer, thanks for the BBQ's you organised at your home and the other various events (sorry I never came to your movie nights). You were also a great Teaching Assistant (TA) for the Machine Learning (ML) course (but do you just show up?). Bidan Huang, we had a great time at ROBIO in China and I am glad to have tasted the hot pot with cow lungs and other various non-prime meat cuts. Snake will have to be for next time. Hang Yin, you hail from the same city in China (Shenyang) as my relatives so we should at least be blood brothers. Although you have a reserved personality we managed to meet up in Shenyang and had an enjoyable time. Ravin Luis, thanks for your knowledge and insight regarding grasping. I was very happy when you told me you enjoyed the lecture I gave on reinforcement learning. Luka Lukic, you could have been a "bouncer" and you had a great sense of humor, I appreciate the time we spent together. Sylvain Calinon, we had great fun at Humanoids and your drinking stamina is impressive. You go to so many conference it might be best you get your liver checked.

In terms of the "newer" arrivals in the lab they proved to be equally or even more diverse in terms of personality. Mahdi Khoramshahi, thanks for providing the backup music to my random lyrics and being my photographer. I am sure we will still get the opportunity to sing about kippahs. Nadia, you are a lot of fun, helpful and down to earth. You really helped me a lot when I first started to use the KUKA and you stayed many times late at night to help me. We spend great time together and we both enjoyed being TAs which made us "complice". Just try to no be broke all the time. Seyed Sina, you work way too much but I am glad you are engaged. You are always ready to help whenever and for whatever. I appreciate the time we spent together and the many political conversations we had. Nili Krausz, you were quite at first but then revealed yourself to be the definition of "chatterbox". Hope you manage to publish your novel and that I get that promised kippah. I was (still) reserved, but I did enjoy having you in the lab as it induced a lot of dynamism. Iason Batzianoulis, I enjoyed the conversations we had regarding Europe and politics in general. I am partially glad I was exempt from your 3-4h experiment. João Abrantes, you were very fun and a legendary TA and Tinder tutor. I enjoyed the discussions we had and the course we took together. I can only hope to get as ripped as you in the future. Ajung Moon, we spent long hours together in "the dungeon" late in the evening. I am glad that you managed to use my code and got to do your experiment. You were always positive and smiling,

although your google calendar schedule would give anybody else nightmares. Laura Cohen, when you joined the lab you brought some aspects of Paris with you. We had great times during the lab long marathons of weekly Friday BBQs. When I lost my motivation your suggestion of "Kaamelott" really cheered me up. Jode Ramon, your relaxed Spanish personality was comforting and I enjoyed the belief-space planning conversations we had. I owe you a bottle of Rosé any day. Mustafa Suphi, we first met when we both gave presentations in LASA before joining. I am glad you joined as you are eager to talk politics which I do as well. It was captivating as we have different political orientations which made our discussions even more interesting. I learned a lot from you and wish you all the best. Felix Duvallet, your joining of the lab and de facto introduction of github was long overdue. I am grateful for your help in introducing me to travis. I was also very uplifted when you told me "good" when my journal got rejected which I immediately understood as words of support. I also appreciate the tips you gave me in scientific writing. Denys Lamotte, we got to know each other as TAs. When I manage to find you we did good work together both for the course and in town. Joel Rey, you are the wisest amongst us. I enjoyed being a TA with you. Jordi Bautista, thanks for the Spanish treats you sent to the lab. Puckett Ashley, thank you for giving me website design advice you are a talented artist.

To all the newbies: Murali, Wissam and Kevin, Godspeed! You are all of reputable character.

A finally would like to deeply thank my wife, Jing, who loves me unconditionally. Thanks to your deep passion for traveling we managed to go to many interesting and exotic holiday destinations. You provided a lot of support to me, I am internally grateful. I wish to thank my parents and family who helped my throughout my Thesis for spell checking this document.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Taking long term decisions or spontaneous reactive actions when presented with incomplete information or partial knowledge is paramount to the survival of any biological or synthetic entity. Reasoning given a state of uncertainty is a continuously occurring event throughout our livelihood. When considering long term decisions an abundance of examples come to mind. For instance, in economic investments uncertainty is to the best of efforts quantified and minimised in order to avoid unwarranted risks. Reactive actions are just as common; when looking for the snooze button of an alarm clock, early in the morning, our hand seems to autonomously search the surrounding space picking up sensory cues gradually acquiring information guiding us towards the button. All the above types of decision require the integration of evidence and an ability to predict the outcomes of the taken decisions in order to ensure a favourable end state. In Artificial Intelligence (AI) and robotics, the ability to reason whilst taking uncertainty into consideration has resulted in mixed levels of success.

There has been noticeable success in artificial agents beating humans at board games such as backgammon (TD-Backgammon), chess (Deep blue) and now recently go (AlphaGo). The gap between robotic autonomous systems and humans starts to diverge however when the action space is continuous and uncertainty is non-negligible. Although there are recent examples of robots coping with such conditions such as opening doors (pose of the handle's position and shape uncertainty of the handle), walking downstairs (state transition uncertainty)(Asimo), and turning valves (DARPA Robotic challenge, DRC Johnson and et. al (2015)), repetition and reproducibility of such behaviour is hard. This was highlighted in the results of the 2015 DRC in which issues (perception, control, software engineering, etc...) resulted in many robots losing balance and falling. [1]

There is an increasing number of robotic application domains where perception is limited, such as planetary[2] and deep water exploration, and where optimal decisions taking uncertainty into consideration play a critical role in

[1] http://www.cs.cmu.edu/~cga/drc/
[2] http://exploration.esa.int/mars/

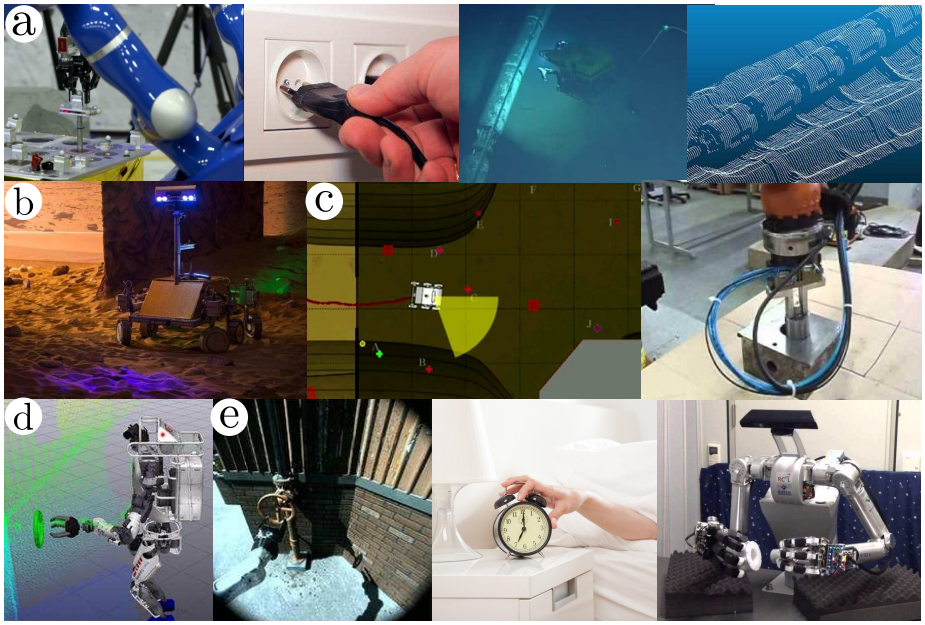


Figure 1.1: Examples of the decision making under uncertainty in both robotics and everyday life situations. (a) European Space Agency (ESA), remote orbital peg in hole task. (b)-(c) ESA, simulated exploration of a cave on Mars in the dark. (d)-(e) MIT DAC team, Atlas robot doing valve task, http://drc.mit.edu/. Other pictures include underwater exploration and industrial peg-in-hole assembly. Bottom-right, a robot equipped with allegro-hands carries out a peg-in-hole assembly ask (Robot Cognition & Control Lab, KITECH)

the success of the tasks undertaken. It would be advantageous to leverage human decision and action abilities for tasks in which the effect of uncertainty is problematic, such as exploration, search and manipulation.

It is not yet fully understood how decisions are taken, yet alone under uncertainty. The difficulty is that two processes responsible for the synthesis of our actions and decisions, that is our beliefs and desires, are not directly or easily measurable. There is growing interest in Neuroscience to understand the mechanisms underlying perception and decision making under uncertainty, (Preuschoff et al., 2013); there is not yet a consensus on the biological mechanisms involved in decision making and efforts are ongoing[3] to construct plausible models of our decision processes. However, seemingly as a result of our prior knowledge and experience, we are better than current robotic systems at handling and dealing with uncertainty. Exploiting human abilities to accomplish tasks in which uncertainty is problematic can help in improving AI algorithms.

AI & robotics considered early on uncertainty in decision making, where the predominant domain of application was spatial navigation, (Cassandra et al., 1996). In these early applications, routes were planned from a start to a goal state, through heuristic methods which chose paths that balanced the reduction in uncertainty and distance taken to reach the goal. The above navigation problem has typically been treated in two parts: the construction and repre-

[3]the human brain project: https://www.humanbrainproject.eu/

sentation of a world model (the map) and a planner which can reason with respect to this model in order to accomplish the objective. The world construction problem attracted a large amount of interest and has resulted in many successfully applications in a wide spectrum of robotic domains (Autonomous Underwater Vehicle AUV, Unmanned Aerial Vehicle UAV, etc...). However, the most successful mapping algorithms are well suited to situations in which a direct observation exists between the robot and the features of the map which is being built and the uncertainty originates from Gaussian noise corrupting the measurement. This assumption breaks down in tasks in which mostly negative information is present, that is the absence of sighting of a feature, such as when exploring a dark cave (Figure 1.1 (b)-(c)) or in environments in which landmarks are sparse or if insufficient sensory information is available such as in haptic and tactile searches.

The integration of planning with mapping in a single framework is still difficult to achieve and is based on either representing the decision problem as a Partially Observable Markov Decision Process (POMDP) which is notoriously difficult to solve for large scale problems or by search heuristics.

The main difficulty faced by planners is that the dimensionality of the state space and decision time horizon leads to an unmanageable space and time complexity optimisation problem. Most data driven optimisation methods such as Reinforcement Learning make the strong assumption that simple explorative strategies (white noise) are sufficient to find optimal decision rules in a relatively short time. This assumption is no longer valid in continuous POMDPs when the number of parameters of the policy is quite large. We can take advantage of expert knowledge from human teachers who can provide a set of explorative and exploitative actions so that although the optimisation problem is large there is no need to perform expensive and time consuming autonomous explorations to find an optimal policy.

In summary there are still open problems in decision making when considering partial observability, which originate from both how decisions are planned and how a map is constructed. As both humans and animals are far better at navigation than robots, especially when uncertainty is present, (Stankiewicz et al., 2006), we decide to leverage human foresight and reasoning in a Programming by Demonstration (PbD) framework (Billard et al., 2008), which we coin PbD-POMDP. PbD examples include the transfer of kinematic task constraints, stiffness and impedance constraints and motion primitives, to name only a few. As for the mapping problem, it has been studied and solved within a certain set of constraining assumptions which do not hold when "negative information" is present (the absence of a positive sighting of a landmark, a term used in the Simultaneous Localisation and Mapping literature), in the case for haptic and tactile search tasks. For the mapping problem we develop a Bayesian filter which is non-parametric and has no explicit parameterisation of a joint distribution values.

In this thesis we address both mapping and planning problems under levels of uncertainty in which no assumption of prior structure, such as Gaussianity, can be assumed.

1.2 Contribution

In this thesis we bring to light three contributions:

1.2.1 Learning to reason with uncertainty as humans.

The first contribution is the transfer of human behaviour to robots, by learning a policy extracted from human demonstrations, in tasks where there is much uncertainty, making them difficult to solve using traditional techniques.

1.2.2 Reinforcement learning in belief space.

The second contribution is an extension of the first contribution, learning to reason with uncertainty as humans. We added a cost function which we demonstrate can be used to refine and improve an original policy solely learned from human demonstrations without any additional simulation or rollouts, in a Reinforcement Learning framework in belief space.

1.2.3 Non-parametric Bayesian state space filter.

The previous two contributions are part of a localisation problem where only the position of the human or robot is unknown. The third contribution addresses the problem when the map of the environment is also unknown and only sparse sensory information is available making traditional mapping and localisation methods inapplicable. We developed a non-parametric Bayesian state space filter which can efficiently handle non-Gaussian joint distributions.

Throughout this thesis we consider case studies in which visual input is not available, leaving only tactile and haptic information. This choice effectively induces a high level of uncertainty as the field of perception is greatly reduced increasing sensing ambiguities making it easier to study its effect on the decision making process. As a consequence the tasks we consider are by nature, haptic and tactile searches. The following three sections detail the contribution of this thesis to research decision making under severe uncertainty constraints.

1.2.1 Learning to reason with uncertainty as humans

A Markov Decision Process (MDP) allows the formulation of a decision problem in terms of states, actions, a discount factor and a cost function. Given this formulation and a suitable optimisation method (dynamic programming, temporal difference, etc..) optimal actions are inferred for each state which are

encoded in a policy. The benefit of this approach is that the policy is non-myopic and sequences of complicated actions can be synthesised to achieve a goal which an opportunistic policy would fail to achieve. A Partially Observable Markov Decision Process (POMDP) is a generalisation of a MDP to a hidden state space in which the state is only observable through measurements. Finding an exact optimal solution to a POMDP problem is notoriously difficult due to the computational complexities involved. Sample based approaches to solve a POMDP rely heavily on a good trade-off between exploration and exploitation actions. Good explorative actions increase the chance of discovering a set of optimal decisions/actions.

In this thesis we propose a Programming from Demonstration approach to solving POMDP problems, which we call PbD-POMDP, in haptic and tactile search tasks. Our hypothesis is that if we know the cognitive map of the human expert in terms of his believed location and observe his actions we can learn a statistical policy which mimics his behaviour. Since the human's beliefs are not directly observable we infer them by assuming that the way we integrate evidence is similar to a Bayesian filter. There is evidence both in cognitive and neuroscience that this is the case (Baker et al., 2011). From observing the expert human performing a task we learn a cognitive model of the human's decision process by learning a generative joint distribution over his beliefs and actions. The generative distribution is then used as a control policy. By this approach we are able to have a policy which can handle uncertainty similarly to humans.

1.2.2 Reinforcement learning in belief space

Learning to search and act as humans and thus reproduce their exploratory behaviour is beneficial in POMDP tasks, since traditional approaches are infeasible. The drawback of the PbD-POMDP approach is that the goal of the task is implicitly encoded in the demonstrations performed by the teacher. To be successful, it is usually required that the teacher is an expert, with a few notable exceptions, (Rai et al., 2013). As a result the quality of the learned behaviour depends on the skill and embodiment constraints of the human. Since we are solely learning a PbD-POMDP statistical controller, both good and bad demonstrations are mixed in together. By introducing a cost function representing the task, we can explicitly obtain a quality metric of the provided demonstrations. In this way we can estimate the parameters of our generative model to optimise the cost function.

Reinforcement learning (RL) is a framework which allows, through repeated interaction with the environment, to learn an optimal policy for a task. There are many variants of RL, but all rely on simple exploration strategies to find the optimal behaviour. These explorative strategies prohibit the application of RL to large and continuous POMDPs in which the policy is comprised of

many parameters. In our previous contribution we showed that it is feasible to learn and extract multiple search strategies from human demonstrations and in a sense we have already solved the exploration/exploitation dilemma which plagues reinforcement learning applications.

We propose a Reinforcement Learning framework for the task of searching and connecting a power plug to a socket, with only haptic information. A set of human teachers demonstrate the task from which we record and build a statistical controller. With the same data we learn a belief space value function which we use to update the parameters of the original statistical controller. In this RL-PbD-POMDP setup a very simple cost function provides a significant policy improvement.

1.2.3 Non-parametric Bayesian state space filter

In both previous contributions we considered searches which can be categorised as localisation problems. In localisation problems the map of the environment is considered to be known while the position of the agent is unknown. There is a wide range of applications for localisation but there are also cases in which both the map and the agent's position is unknown. This kind of problem is known as Simultaneous Localisation and Mapping (SLAM).

SLAM is concerned with the development of filters to accurately and efficiently infer the state parameters of an agent (position, orientation) and aspects of its environment, commonly referred to as the map. It is necessary for the agent to achieve situatedness which is a precondition to planning and reasoning. The predominant assumption in most applications of SLAM algorithms is that uncertainty is related to the noise in the sensor measurements. In our haptic search tasks there is no visual information and a very large amount of uncertainty. Most of the sensory feedback is negative information, a term used to denote the non-event of a sensory response. In the absence of recurrent sightings or direct measurements of objects there are no correlations from the measurement errors which can be exploited.

In this thesis we propose a new SLAM filter, which we name Measurement Likelihood Memory Filter (MLMF), in which no assumptions are made with respect to the shape of the uncertainty (it can be Gaussian, multi-modal, uniform, etc..) and motion noise and we adopt a histogram parametrisation. The conceptual difference between the MLMF and standard SLAM filters, such as the Extended Kalman Filter (EKF), is that we avoid representing the joint distribution since it would entail a unfathomable space and time complexity. This is achieved by keeping track of the history of measurement likelihood functions. We demonstrate that our approach gives the same filtered marginals as a histogram filter. In such a way we achieve a Bayes filter which has both linear space and time complexity. This filter is well suited to tasks in which the landmarks

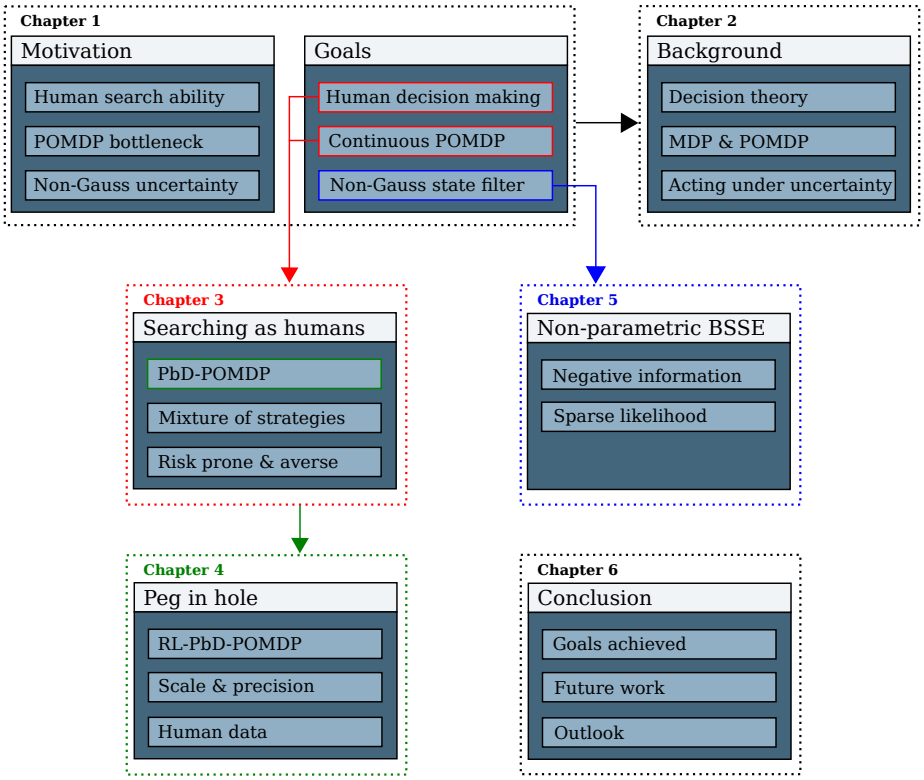


Figure 1.2: Roadmap of the Thesis with. Three contributions are partitioned in the 3rd, 4th and 5th chapters.

are not directly observable.

1.3 Thesis outline

The thesis is structured according to three main contributions outlined in the previous section, each comprising a chapter and the following paragraphs give a detailed outline of the structure of this thesis, see Figure 1.2.

Chapter 2 - Background

In this chapter we introduce and mathematically formalise the sequential decision making problem under uncertainty and we provide a detailed literature review of the related work in this domain. We provide a brief introduction to *Decision Theory* before focusing on the work in AI & robotics relevant to POMDPs whilst highlighting their relevance and contribution to our work.

Chapter 3 - Learning to reason with uncertainty as humans

In this chapter we present an approach for transferring human skills in a blind haptic search task to a robot in our PbD-POMDP framework. The belief of the human is represented by a particle filter and all subsequent beliefs are inferred from the human's motions acquired via a motion tracking system. A generative model of the joint belief and actions distribution is learned and used to reproduce the behaviour on a WAM and KUKA robot in two search tasks. Experimental evaluations showed the approach to be superior to greedy opportunistic policies and traditional path planning algorithms. We also provide a review of work related to humans taking decisions under uncertainty in spatial navigation and haptic tasks with an emphasis on works which consider diminished or no visual information. This chapter has been published in de Chambrier and Billard (2013, 2014).

Chapter 4 - Reinforcement learning in belief space

In this chapter we present a similar approach to the one in Chapter 3, "Learning to reason with uncertainty as humans", with the difference that we explicitly encode the task through the introduction of a binary objective function and we consider a peg-in-hole task under high levels of uncertainty. The task requires both high and low levels of precision from the agent to be able to accomplish it, which makes it particularly interesting. We demonstrate the importance of initially provided human data as opposed to using data generated from a greedy policy. We learn a value function approximation of the belief space through locally weighted regression and approximate dynamic programming. By combining a PbD approach in this Actor-critic Reinforcement Learning framework, we demonstrate an improvement upon a purely statistical controller with nearly no additional cost. We refer to this approach as RL-PbD-POMDP. Elements of this chapter are to appear in de Chambrier and Billard (2016a).

Chapter 5 - Non-parametric Bayesian state space filter

In this chapter we present an approach to perform a state space estimation of a map and agent given that there is no direct observation between the landmarks and the agent. We demonstrate that by keeping track of the applied measurement functions rather than explicitly parametrizing the full joint distribution of the landmarks and agent we can fully reconstruct the optimal Bayesian state estimation. The advantage of our approach is that the space complexity is linear as opposed to exponential. We validate our approach in 2D search navigation tasks. We also give an overview of the literature of SLAM and elucidate the position of our filter within it. This chapter has been submitted to de Chambrier and Billard (2016b) and is under review.

Chapter 6 - Conclusion

We conclude by providing a holistic summary of our work and achievements. We draw attention to the current open problems and directions for future work in the field of uncertainty and reasoning in AI and robotics.

Chapter 2

Background

Acting under uncertainty is central to AI and robotics and has been an active area of research for decades. It is an umbrella term which encompasses a wide spectrum of fields: *economics*, *psychology*, *cognitive science*, *neuroscience*, *robotics* and *artificial intelligence*. The work in this thesis relies on results from all of the aforementioned fields with varying degree. Cognitive and neuroscience bring justification and biological inspiration in the way we represent our beliefs and how we act accordingly. AI and robotics provide computational models and optimisation methods some of which are biologically inspired to be able to solve decision problems given uncertainty. Because of the vast spectrum of topics we cannot do justice to all them and we will focus on works which are directly relevant to the problems we are addressing in this thesis, *which is how to teach a robotic apprentice to act under uncertainty*. In this chapter we cover the following topics in the presented order: Decision Theory (DT), Markov Decision Process (MDP), Partially Observable Markov Decision Process (POMDP), a literature review and the approach taken in this thesis.

Figure 2.1: Chapter outline.

- **Section** 2.1, introduces what is meant by taking decisions under uncertainty and what are the different sources of uncertainty. We take a historical look at Decision Theory since it is the root node of all subsequent research in reasoning and acting under uncertainty and provides for a good introduction to the topics which will follow.

- **Section** 2.2, mathematically formalises the sequential decision problem under uncertainty and is linked with Decision Theory. We derive from first principles the Bellman optimal equation which is one of the most important results to date in sequential decision processes.

- **Section** 2.3, provides an in depth literature review with the latest results in AI and robotics in the subject of planning and acting under uncertainty.

We draw attention to the different approaches to solving this problem whilst pointing out their advantages and weaknesses. We summarise what has been achieved so far and what are the open problems.

- **Section** 2.4, the core approach taken by this thesis is detailed. We outline how we teach a robotic apprentice to act under uncertainty.

2.1 Decisions under uncertainty

The main objective of reasoning under uncertainty is to find an action or sequence of actions which will result in the most preferable outcome. There are two key attributes which can render this problem difficult: **stochastic actions** and **latent states**.

Stochastic actions when applied in the same state will not always result in the same outcome. This type of uncertainty can arise from many sources. For instance, the outcome of random systems will always lead to different results when the same action is applied to the same initial conditions, such as the throwing of a dice or the flipping of a coin. In outdoor robotics the terrain might lead to slippage, causing the robot to skid, or in an underwater environment currents might drastically offset the position of an AUV. In articulated robots, the friction in the joints can result in an error in the end-effector position (especially true for cable driven robots).

The second source of uncertainty is when the state space cannot be accurately estimated. This arises when the sensors are not able to provide sufficient information to reliably estimate the state. In robotics this uncertainty can arise from inadequate or noisy sensors. In poor environmental conditions such as humidity, lack of light or smoke the robot can experience difficulties in ascertaining its position and thus in planning how to achieve a given objective.

Given these two types of uncertainty, the question is how to represent these uncertainties. The predominant approach is to quantify the uncertainty in terms of probabilities. For instance the application of a forward action to a wheeled robot will result in some probability in a new position further ahead and with a remaining probability distributed to adjacent regions which might have occurred due to slippage.

An observation made through the robot's sensors will result in a probability distribution over the robot's probable location. This quantification of the action and observation uncertainty, in terms of a probability distribution over the state, must be utilised by the agent to plan actions towards accomplishing its goal. In order to take a decision, the agent must assign a utility to each state weighted by the probability of its outcome and act so as to get the highest utility. The utility indicates a preference over the outcomes and when combined with probabilities leads to Decision Theory, which is the topic of the next section.

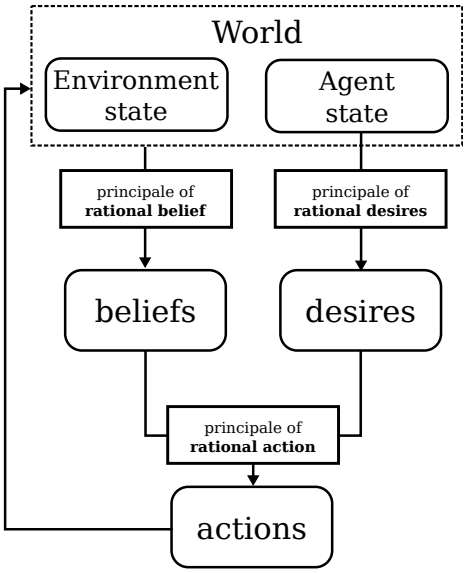


Figure 2.2: Relation between beliefs, desires and actions and are all considered to be rational.

2.1.1 Decision theory

The central question that Decision Theory asks is: *how do we take decisions when faced with uncertain outcomes ?* To answer such a question we need to identify the attributes which are involved when we take a decision, namely our **beliefs** and **desires**. Beliefs reflect a degree of knowledge we have about the world. This degree is ascertained by the amount of evidence we have in support of our beliefs. Epistemology studies in great detail the relationship between truth, beliefs and knowledge. We will not go into a philosophical discussion of their interplay, but make use of the following: if we have sufficient evidence in support of our beliefs and they represent the truth then we consider them to be **rational beliefs**. As for desires, they are linked to our disposition to take upon them. For example if I want to switch off my alarm clock I have to look for it in the last area I believed it to be in. These two attributes, beliefs and desires, are used to frame a decision problem. Early work in decision theory assumed that the problem was well grounded and focused on finding what **rational actions** need to be taken given our beliefs in order to achieve our desires. For a detailed review on Decision Theory the reader is referred to Steele and Stefansson (2015); North (1968)

Early interest in such questions were typically centred around economics which included deciding an appropriate investment or wager for a particular gamble. It was noted that the expected monetary outcome of a gamble as a means of basing a decision, would often lead to a course of action which contradicts common sense. A famous example of this contradiction is demonstrated in the St. Petersburg paradox. In this paradox a bookmaker proposes the follow-

ing gamble. An initial pot starts with a content of £2. The bookmaker proceeds to flip a fair coin until the first appearance of a tails which ends the game. Until the occurrence of the first tails the money in the pot doubles after every toss. Once the game ends the player leaves with the contents of the pot. As an avid gambler and **expected value** maximiser how much would one be willing to pay to enter this game ? To access, one would need to know the average payout. The amount of money increases by $£2^n$, where n is the number of non-final tosses and the probability of reaching n is $1/2^n$. In this case the expected monetary outcome is infinite:

$$\underset{p(£)}{\mathbb{E}}\{£\} = \underbrace{\frac{1}{2}£2}_{\text{first toss}} + \frac{1}{4}£4 + \cdots = \sum_{n=1}^{\infty} £\frac{2^n}{2^n} = £\infty$$

So the expected gain or return for paying to enter such game is an infinite amount of money. Thus in principle if a player was seeking to maximise his expected return value he would be willing to pay an amount close to infinity to enter the game. This does not seem a good decision rule; no person in the world would be willing to pay such high amounts to enter this game.

Nicolas Bernoulli proposed a solution to the problem which was later published by his brother Daniel (republished Bernoulli (1954)). He introduced the notion of a **utility function**, and he claimed that people should base their decision on the expected utility instead of solely on the monetary outcome.

> "...the value of an item must not be based on its price, but rather on the utility it yields."
>
> — Daniel Bernoulli

The introduction of a utility function takes into account that the net worth of a person will influence their decision since different people (in terms of their monetary worth) will weigh the gain differently. The utility function introduced by Bernoulli was the logarithm of the monetary outcome $x \in X$ weighted by its probability $p(x)$ which results in an expected utility:

$$U(x) = \mathbb{E}\{u(x)\} = \sum_{x \in X} p(x) \underbrace{\log(x)}_{u(x)}$$

It is later in 1944 that von Neumann and Morgenstern (vNM)(von Neumann and Morgenstern, 1990) axiomised Bernoulli's utility function and proved that if a decision maker has a preference over a set of lotteries[1] which satisfy four axioms (completeness, transitivity, continuity, independence) then there exists a utility function whose expectation preserves this preference. An agent whose decisions can be shown to maximise the vNM expected utility are said to be **rational** otherwise they are **irrational**.

[1] the term lottery refers to a probability distribution in the original text.

This is the theoretical basis of most economic theory. It is a **normative** model of how people should behave given uncertainty. It is also the basis of most if not all decision making, cognitive architectures and control policies in AI and robotics (to the best of the author's knowledge).

An aspect to keep in mind regarding the vNM model is that it is normative; it states what should be a rational decision. As a result it is not always consistent with human behaviour. There is great debate regarding the predictions made by vNM models with respect to our behaviour. There have been many studies both demonstrating divergence between the model's predictions and our observed behaviour but also supporting evidence that it does reflect the output of our decision making process. Reasons for divergence have been attributed to how people weigh probabilities and how the decision problem is framed. But probably the most important aspect is that in most decisions we are faced with, the quantification and rationality of our beliefs might not be adequate and limitations of our working memory will come into play in the final decision.

Nevertheless vNM agents are predominantly used in AI and robotics as a means of implementing decision making processes or in control policies. In psychology and cognitive science vNM agents are a used for comparing human behaviour against an optimal strategy (by optimal we mean it is rational in the vNM sense). It is important to remember the origins and assumptions underlying the models that are used to represent control policies or cognitive architectures implemented into robotic systems or software agents.

2.2 Sequential decision making

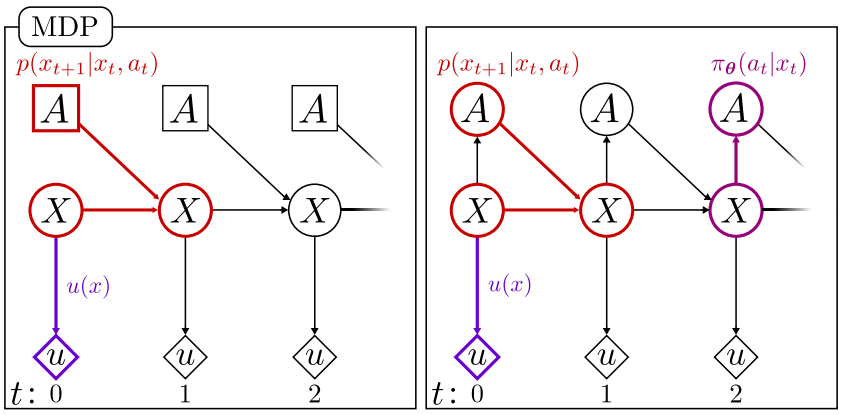
When Decision Theory is brought up, we are usually referring to a one shot non-temporal decision. However many interesting decision problems are sequential. In such situations, we must consider the effect current decisions will have on future decisions. Expected utility theory (part of Decision Theory) is extendable to a temporal decision problem. There are however two subtle but important differences between the temporal and non-temporal decision problems. The first difference is the utility. In the one time step problem an outcome has one utility assigned to it, $u(x)$. In the temporal decision problem a utility has to be assigned to a sequence of outcomes, $u(x_{0:T})$, where T is the number of sequential decisions taken. The utility of a sequence is the sum of the individual utilities. However if the decision problem is non terminating this will lead to an unbounded utility. To bound the utility a discount factor $\gamma \in [0, 1)$ is introduced and the new temporal utility function becomes:

$$u(x_{0:T}) := \sum_{t=0}^{T} \gamma^t u(x_t) \tag{2.2.1}$$

Notation	Definitions
$x_t \in \mathbb{R}^3$	Cartesian state space position of the agent end-effector.
$y_t \in \mathbb{R}^M$	Observation/measurement from the agents sensors.
$a_t \in \mathbb{R}^3$	Action, Cartesian velocity of the end-effector of the agent.
X, Y, A	State, observation and action random variables where x, y and a are realisation.
$p(x_t)$	Short hand notation for a probability density function, $p_X(x_t)$.
$x_{0:t}$	$\{x_0, x_1, \cdots, x_{t-1}, x_t\}$, history up to time t.
$p(x_t \vert y_{0:t}, a_{1:t})$	Filtered probability distribution over the state space given the action and observation history.
b_t	Belief state is the filtered state space distribution $b_t = p(x_t \vert y_{0:t}, a_{1:t})$ which will be written as b_t for simplicity.
$\pi_{\boldsymbol{\theta}}(a_t \vert \cdot)$	Parametric probabilistic policy, $a_t \sim \pi_{\boldsymbol{\theta}}(a_t \vert \cdot)$, where $\boldsymbol{\theta}$ is the parameters.
$u(x) \in \mathbb{R}$	Utility function, returns the utility of being in state x. It can also be dependent on the action, $u(x, a)$.
$\gamma \in [0, 1)$	Discount factor, the closer to one the more the later utilities are considered. When set to zero, only immediate utilities are considered which would result in a myopic greedy agent.
$p(x_{t+1} \vert x_t, a_t)$	State transition model, returns the likelihood/probability of reaching state x_{t+1} given that action a_t is applied in state x_t.
$p(y_t \vert x_t)$	Observation/measurement model, returns the likelihood/probability of observing y_t given that the agent is in state x_t.
$\tau(b_t, u_t, y_t)$	Updates a belief given a motion and observation. It makes use of both the motion and observation functions. The state space estimation function, τ, can be any kind of state space filter such as an Extended Kalman Filter (EKF) or a Particle Filter (PF).

Table 2.1: Definition of common variables used.

The discount factor controls the importance that later utilities have on the final utility. If the discount factor is set to zero we obtain the original one shot utility function and if we were to take actions which maximised the expected utility we would not be considering at all the effect current decisions have at future decision points. An agent reasoning in such a way is called **myopic**. The second difference between the temporal and non-temporal decision problem is the way in which probabilities are assigned to outcomes. This was $p(x)$ in the Decision Theory utility function formulation. Now because of the sequential nature of the problem we consider a conditional state transfer probability distribution $p(x_{t+1}|x_t, a_t)$ which models the probability of going from state x_t to x_{t+1} given that action a_t is taken. This particular representation of a sequential decision problem is called a **Markov Decision Process (MDP)** and to be more exact a first order MDP. The necessary models are the state transition and utility functions. The assumption of such a model is that all necessary information to take a decision is encoded in the current state and there is no need to consider the history of state transitions when taking a current decision. In Figure 2.3 we illustrate two graphical representations of a MDP, which are known as **Dynamic Bayesian Networks (DBN)**. A DBN represents the temporal relationship and conditional dependence between random variables, decisions and utilities, which are represented by circles, squares and diamonds. For the MDP to the left the actions are not stochastic, whilst for the MDP on the right the actions taken are governed by a stochastic **policy**, $\pi_{\boldsymbol{\theta}}(a_t|x_t)$. A policy represents the plan of an agent for each state, given a state it will output an action. A policy is considered optimal when it maximises the expected utility function, it is optimal in the vNM sense.



(a) Off-policy DBN **(b)** On-policy DBN

Figure 2.3: Dynamical Bayesian Network of a Markov Decision Process; it encodes the temporal relation between the random variables (circles), utilities (diamond) and decisions (squares). The arrows specify conditional distributions. In **(a)** the decision nodes are not considered random variables whilst in **(b)** they are. From these two DBN we can read off two conditional distributions, the state transition distribution (in red) and the action distribution (in purple).

Solving a MDP means finding a policy whose actions in any given state

will always maximise the expected utility. Such a policy is usually denoted as π^*, the **optimal policy**. As in decision theory, the expected utility is the utility of a sequence of states $u(x_{0:T})$ weighted by its probability. The graphical representation (Figure 2.3 (a)) allows the probability of a sequence of states and actions, to be read off directly:

$$p(x_{0:T}, a_{0:T-1}) = p(x_0) \prod_{t=0}^{T-1} p(x_{t+1}|x_t, a_t) \tag{2.2.2}$$

$$u(x_{0:T}) = u(x_0) + \gamma u(x_1) + \cdots + \gamma^{T-1} u(x_{T-1}) + \gamma^T u(x_T) \tag{2.2.3}$$

We are interested in finding the sequence of actions, $a_{0:T}$, which will maximise the expected utility function:

$$\underset{a_{0:T-1}}{\operatorname{argmax}} U(x_{0:T}, a_{0:T-1}) = \max_{a_0} \sum_{x_1} \cdots \max_{a_{T-1}} \sum_{x_T} \left(p(x_{0:T}, a_{0:T-1}) u(x_{0:T}) \right) \tag{2.2.4}$$

Solving the above directly in its current form would to lead to an exponential complexity. Making use of the first order Markov assumption and that current utilities do not depend on future utilities, the summations can be re-arranged and a recursive pattern emerges which can be exploited:

$$\underset{a_{0:T-1}}{\operatorname{argmax}} U(x_{0:T}, a_{0:T-1}) = \max_{a_0} \sum_{x_1} \cdots \max_{a_{T-2}} \sum_{x_{T-1}} p(x_{0:T-1}, a_{0:T-2})$$
$$\left(u(x_{0:T-2}) + \gamma^{T-1} \left(u(x_{T-1}) + \gamma \max_{a_{T-1}} \sum_{x_T} p(x_T|x_{T-1}, a_{T-1}) u(x_T) \right) \right) \tag{2.2.5}$$

From the rearrangement we notice that Equation 2.2.5 has the same functional form as Equation 2.2.4, except that the recursive component can be summarised by Equation 2.2.6, which is known as the **Bellman** optimal equation (the asterisk indicating that it is optimal),

$$V^*(x_t) := u(x_t) + \gamma \max_{a_t} \sum_{x_{t+1}} p(x_{t+1}|x_t, a_t) V(x_{t+1}) \tag{2.2.6}$$

where for the terminal state $V_T(x_T) = u(x_T)$. The Bellman equation is a means of solving a sequential decision problem through use of dynamic programming. It shows that the utility of the current state is based on the immediate utility and the discounted maximum utility of the next state. Making use of this recursion reduces the computation complexity which is now quadratic in the number of states, $\mathcal{O}(T\,|A|\,|X|^2)$. To find the optimal value and subsequent policy an approach would be to repeatedly apply the Bellman equation to each state until the value function converges. What makes the problem difficult to solve is maximisation over the actions. This induces two problems, the first is that the optimisation is nonlinear and the second is that if the action space is

continuous the maximisation will be expensive to compute. This brings into play the two main approaches to solving a MDP: **off-policy** and **on-policy**. Off-policy methods solve directly for the optimal value function, $V^*(x)$, and perform the maximisation over the actions. **Value-Iteration (VI)** is such a method. On-policy approaches, $V^\pi(x)$, find the optimal value and policy through repeating **policy evaluation** and **improvement** steps. In the policy evaluation the value or utility of a policy is found through solving the on-policy version of the Bellman equation:

$$V^\pi(x_t) := u(x_t) + \gamma \sum_{a_t} \pi_{\boldsymbol{\theta}}(a_t|x_t) \sum_{x_{t+1}} p(x_{t+1}|x_t, a_t) V(x_{t+1}) \tag{2.2.7}$$

In the policy improvement step, the policy is made more greedy by maximising the value function. Through the repetition of these two steps both the value function and policy converge to the optimal. On-policy methods are preferred in settings where the action space is highly continuous, such as in robotics. Using dynamic programming is however not the method of choice since it requires multiple passes through the entire state space and for this reason it is necessary to have the model of the state transition a priori. Instead **Reinforcement Learning (RL)** methods are used to find an optimal value and policy. RL is a sample based approach in which an agent interacts with the environment gathering examples of state transitions and the utility and uses them to gradually solve the Bellman equation.

We introduced the formulation of a sequential decision process for the MDP model and showed how an optimal policy and value function are obtained through maximising the expected utility. The re-arrangement of the summations, known as variable elimination, allows to exploits a recursive structure present in the Markov chain. The recursive component turns out to be the Bellman optimal equation, which when solved (via dynamic programming or reinforcement learning) results in an optimal value and policy function. A MDP models the uncertainty inherent in the state transition but not the uncertainty of the state. The MDP assumes that the state space is always fully observable, which is a strong assumption. In robotics, the on board sensors return an estimate of the state with a certain amount of uncertainty associated with it. To take this additional uncertainty into consideration the MDP has to accommodate it. This leads to a Partially Observable Markov Decision Process (POMDP).

2.2.1 POMDP

A POMDP is a popular approach for formulating a sequential decision process in which both motion and observation uncertainty are considered. In this partially observable setting the agent does not know with exactitude the state

of the environment, but is able to observe it through his **sensors**. We define a sensor mathematically as being a function of the state space, x_t, relating to an observation, y_t, corrupted by some noise, ϵ_t,

$$y_t = h(x_t) + \epsilon_t \tag{2.2.8}$$

The sensor function $h(x_t)$ can be linear or non-linear and the additive noise term ϵ_t can be Gaussian (usually the case), non-Gaussian, state dependent or not. The uncertainty of the latent state, x_t, is quantified by a probability distribution, $p(x)$. This probability distribution represents all the hypothetical positions in the world in which the agent can be found. In Figure 2.4 **(a)** an agent is located in a square yard containing a wall. Initially the agent is confident of his position; his state uncertainty $p(x_0)$ is low, represented by the blue probability density. However during a circular displacement the agent skids and the state uncertainty is increased by the state transition function, $p(x_{t+1}|x_t, a_t)$; this step is referred to as **motion update**. To reduce the uncertainty, the agent takes a measurement, y_t, with his sensors which provide range, r, and bearing, ϕ, information with respect to the wall, see Figure 2.4 **(b)**. The agent uses the model of his sensor, known a priori, to deduce all possible locations in the world from where the current measurement could have originated. This model is known as the measurement likelihood function:

$$p(y_t|x_t) = \mathcal{N}(y_t - h(x_t); 0, \Sigma) \tag{2.2.9}$$

The measurement likelihood function makes use of the measurement function $h(x)$ and it models the noise in the sensor. In this case the noise model, ϵ_t, is Gaussian, paramaterized with mean zero and covariance Σ. Typically the parameters of the measurement likelihood function are learned a priori.

In Figure 2.4 **(c)** the likelihood is illustrated. The dark regions indicate areas of high likelihood, which are possible locations from which the sensor measurement could have originated. The value of the measurement likelihood function is then integrated into the state space probability density function; this step is referred to as **measurement update**.

The two update steps, motion and measurement, are part of a recursive state estimation process called a **Bayesian state space filter**, which we formalise below in Equation 2.2.10-2.2.11.

The motion model, Equation 2.2.10, updates the position of the probability distribution according to the applied action, a_t, and adds uncertainty by increasing the spread of the distribution. The measurement information is then incorporated by Equation 2.2.11. The measurement likelihood always reduces the uncertainty or leaves it constant. The Bayesian state space filter is such an important component to belief space decision making that we define it by the filter function, $\tau(b_t, a_t, y_t)$, which takes as input the current belief, applied action and sensed measurement and returns the resulting belief b_{t+1}. The state

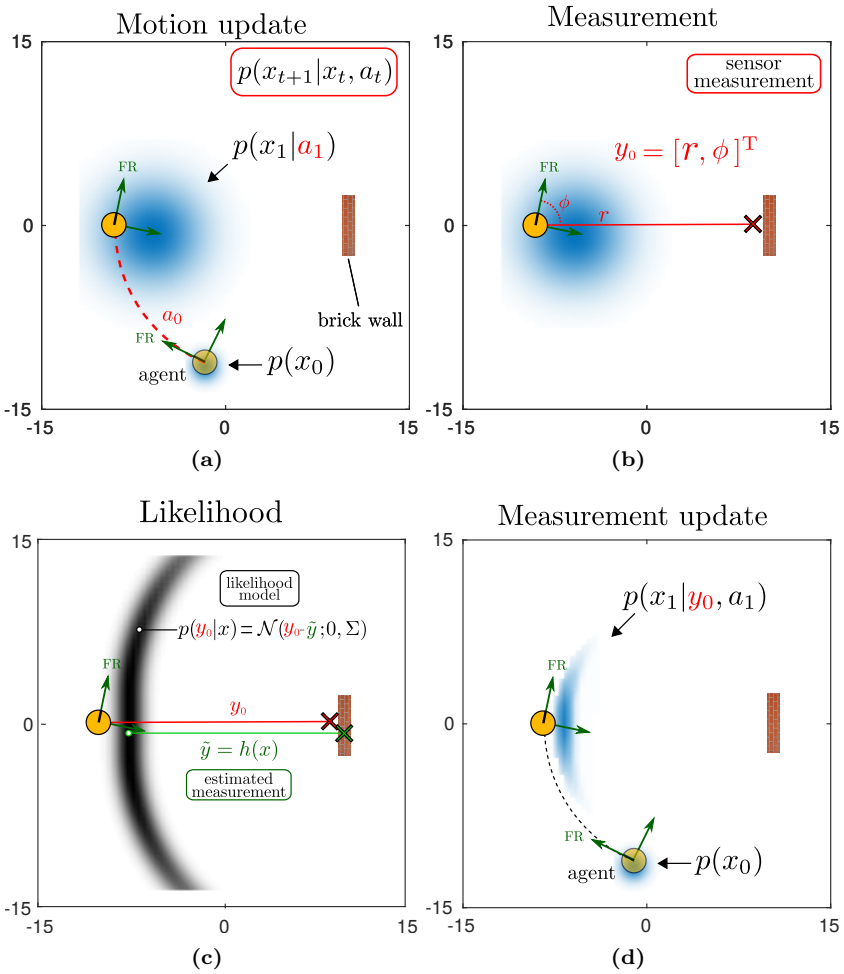


Figure 2.4: **(a)** An agent is located to the south west of a brick wall. It is equipped with a range sensor. The agent takes a forward action but skids, which results in a high increase of the uncertainty.**(b)** The agent takes a measurement, y_0, of this distance to the wall; because his sensor is noisy his estimate is inaccurate. **(c)** The agent uses his measurement model to evaluate the plausibility of all locations in the world which would result in a similar measurement; illustrated by the likelihood function $p(y_0|x_0)$. **(d)** The likelihood is integrated into the probability density function; $p(x_0|y_0) \propto p(y_0|x)p(x_0)$.

Bayesian filter

The Bayesian filter turns a prior probability distribution over the state space, $p(x_{t-1}|y_{0:t-1}, a_{1:t-1})$, to a posterior $p(x_t|y_{0:t}, a_{1:t})$ by incorporating both motion and measurement. Applied recursively it keeps a probability distribution over the state space which considers all the past history of actions and observations. We define the application of these two steps by the filter function τ, which takes the current belief, the applied action and measurement, and outputs the next belief, b_{t+1}.

Motion update

$$p(x_t|y_{0:t-1}, a_{1:t}) = \int p(x_t|x_{t-1}, a_t)\, p(x_{t-1}|y_{0:t-1}, a_{1:t-1})\; dx_{t-1} \tag{2.2.10}$$

Measurement update

$$p(x_t|y_{0:t}, a_{1:t}) = \frac{1}{p(y_t|y_{0:t-1}, a_{1:t})} p(y_t|x_t)\, p(x_t|y_{0:t-1}, a_{1:t}) \tag{2.2.11}$$

$$p(y_t|y_{0:t-1}, a_{1:t}) = \int p(y_t|x_t)\, p(x_t|y_{0:t-1}, a_{1:t}) dx_t \tag{2.2.12}$$

Filter function

$$b_{t+1} := \tau(b_t, a_t, y_t) \tag{2.2.13}$$

Figure 2.5: Bayesian state space filter.

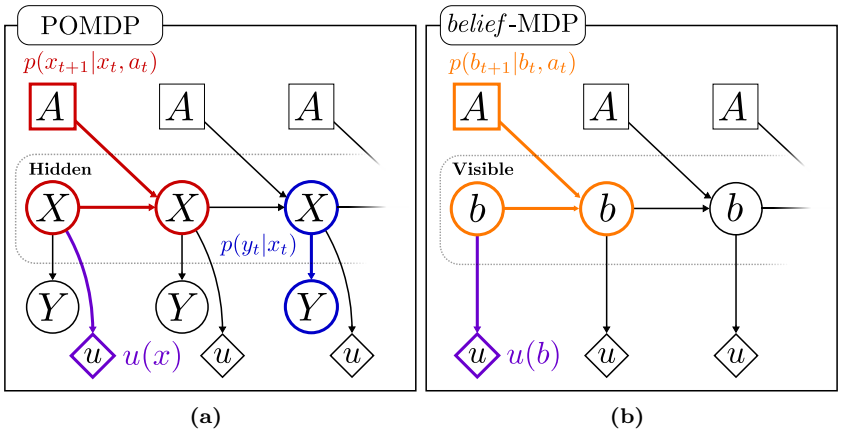


Figure 2.6: **(a)** POMDP graphical model. The state space, X, is hidden, but is still partially observable through a measurement, Y. **(b)** The POMDP is cast into a belief Markov Decision Process, belief-MDP. The state space is a probability distribution, $b(x_t) = p(x_t)$, (known as a belief state) and is no longer considered a latent state. The original state transition function $p(x_{t+1}|x_t, a_t)$ is replaced by a belief state transition, $p(b_{t+1}|b_t, a_t)$. The reward is now a function of the belief.

space filter is an essential component to a POMDP which will become apparent later.

With the latent state, its relation to the observation variable and the Bayesian filter defined, we can introduce the POMDP model in Figure 2.6 (*left*). It has the same Markov chain structure as the MDP, introduced in the previous section, but the state space X is latent and a new layer of observation variables Y is added.

As the state space is only partially observable the expected utility has to be computed for each possible history of states, actions and observations. All approaches in the literature instead encapsulate all these possible histories into a belief state $b(x_t)$ (for short notation b_t) which is a probability distribution (also referred to as an information state, I-state) over the state space x_t and use this new state description to cast the POMDP into a **belief-MDP** (states are probability distributions, beliefs). By casting a POMDP into a *belief*-MDP the state space is considered observable and we recover the same structure as in the standard MDP problem.

As we are working within a belief space the reward function has to be adapted to:

$$u(b_t) = \sum_{x_t} u(x_t)\, b(x_t) = \mathbb{E}_{b_t}\{u(x_t)\} \tag{2.2.14}$$

which is an expectation. The goal as before is to find a sequence of actions which will maximise the expected utility. Since our *belief*-MDP has the same structural form as the MDP, the solution to the problem is the same Bellman equation derived previously. We just substitute the new belief transition function and we

get the corresponding belief Bellman Equation, 2.2.15.

$$V^*(b_t) = u(b_t) + \gamma \max_{a_t} \sum_{b_{t+1}} p(b_{t+1}|b_t, a_t)\, V^*(b_{t+1}) \tag{2.2.15}$$

However, using this equation in this form is problematic, as we are summing over the space of beliefs (which is high dimensional and infinite for the continuous case) and the transition function is a probability distribution over beliefs. The key to overcome this problem is to realise that if we know what the current measurement and applied action are, there is only one resulting belief, b_{t+1}, and the summation over beliefs vanishes. This can be seen by substituting the belief transition function, Equation 2.2.16, into the Bellman equation Equation 2.2.15.

$$p(b_{t+1}|b_t, a_t) = \sum_{y_t} p(b_{t+1}|b_t, a_t, y_t)\, p(y_t|y_{0:t-1}, a_{0:t}) \tag{2.2.16}$$

After the substitution and re-arrangement of the summation we get Equation 2.2.17. Since the observation is known (because the outer summation is over y_t), the summation over the beliefs vanishes since there is only one possible future belief which is given by the Bayesian filter function $b_{t+1} = \tau(b_t, a_t, y_t)$,

$$u(b_t) + \gamma \max_{a_t} \sum_{y_t} \underbrace{\left(\sum_{b_{t+1}} p(b_{t+1}|b_t, a_t, y_t)\, V^*(b_{t+1}) \right)}_{1 \cdot V^*(\tau(b_t, a_t, y_t))} p(y_t|y_{0:t-1}, a_{0:t}) \tag{2.2.17}$$

which simplifies to:

$$\begin{aligned} V^*(b_t) &= u(b_t) + \gamma \max_{a_t} \sum_{y_t} p(y_t|y_{0:t-1}, a_{0:t})\, V^*(\tau(b_t, a_t, y_t)) \\ &= u(b_t) + \gamma \max_{a_t} \mathbb{E}_{y_t}\{V^*(b_{t+1})\} \end{aligned} \tag{2.2.18}$$

The belief Bellman equation is intuitive. The value of the current belief is the immediate utility plus the value of the future belief states weighted by the probability of a measurement which would result in these future belief states. An exact solution exists only when considering a finite state, action and observation space and a finite planning horizon T, Smallwood and Sondik (1973). The belief-MDP can be solved with value iteration but each backup operation (application of the bellman equation) results in an exponential growth in the number of parameters needed to represent the value function, which is computationally intractable.

Most early techniques for solving POMDPs used value iteration. The preference for persisting in doing this, given the computational burden, is that since the utility function uses a linear operator (the expectation) and that the Bellman backup operation (applying the Bellman equation to the current value function) preservers the linearity, the value function after each updates is Piece

Wise Linear and Convex (PWLC). A good text on the implementation of exact value iteration for POMDPs can be found in (Thrun et al., 2005, Chap. 15) and Kaelbling et al. (1998).

In summary there are two problems in solving a POMDP:

- **curse of dimensionality**: A discrete state space of size N will result in a belief space of dimension $N-1$. The discretization choice will greatly impact the computational cost of value iteration.
- **curse of history**: The space and computational complexity in the worst case is exponential with respect to the planning horizon, T (Du et al., 2010).

Given such complexity it is hard to see POMDPs being actually usable for real world scenarios. As a result many approximate techniques have emerged with some being very successful. In the next section, we survey the literature and the developments of approximate POMDP algorithms and their applications.

2.3 Literature review

We review the latest methods on *acting under uncertainty*. This is an extremely dense and spread out area of research, no doubt because of its importance. If uncertainty is not considered by a policy, it risks to result in failure or wast considerable resources during its execution. The review is organised in four subsections:

- 2.3.1 Value Iteration
- 2.3.2 Policy Search
- 2.3.3 Planning
- 2.3.4 Heuristics

with an emphasis on robotic applications. In Figure 2.7 we illustrate graphically these four topics with their associated sub-fields.

2.3.1 Value Iteration

The POMDP formulation introduced previously is the main theoretical starting point of policies which consider uncertainty optimally. However solving an exact POMDP through dynamic programming (value iteration) is computationally intractable and an exact solution only exists for discrete state, action and observation space (Thrun et al., 2005, Chap. 15). This intractability, in which only problems with a few states could be solved has inhibited the application of the POMDP framework to robotics.

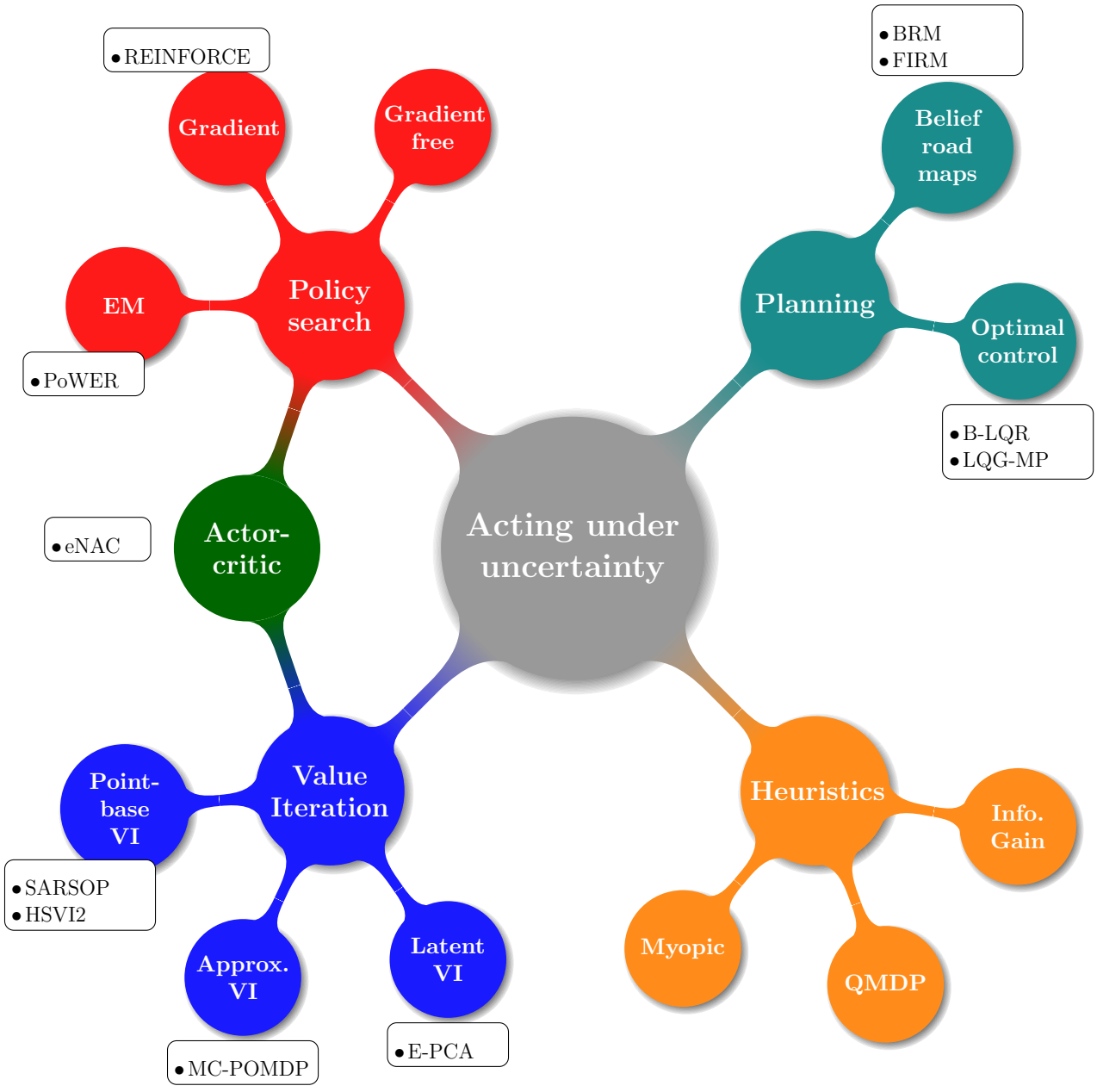


Figure 2.7: Mind-map of AI and robotic methods for acting under uncertainty.

The first breakthrough of the application of Value Iteration (VI) in belief space to a robotic application was Point-Based Value Iteration (PBVI) (Pineau et al., 2003). It allowed VI to be applied to a robotic navigation problem consisting of 626 states in a hospital patient search task. The key insight to scale VI was to only consider a subset of belief states which were reachable and relevant to the problem. This is achieved by smart sampling techniques and only performing VI backups on beliefs states which are relevant. From this point most research has focused on determining efficient strategies to sample belief points and on which to apply VI. Heuristic Search Value Iteration (HSVI1) (Smith and Simmons, 2004) and HSVI2 (Smith and Simmons, 2005) use forward search heuristics to find relevant beliefs by keeping a lower and upper bound on the current estimated value function. The belief tree is expanded by choosing an action and observation with relation to the potential future effect on the value of the bounds, which are being minimised. HSVI has a comparable performance with respect to classical PBVI except in the game of tag (a benchmark problem), in which it fairs significantly better. A method developed after HSVI, named Forward Search Value Iteration (FSVI) (Veloso, 2007) takes an alternative approach to keeping an upper and lower bound on the value function, as in HSVI, since doing so results in a drastic increase in the computation time necessary to find a solution. Instead FSVI assumes that the state space is fully observable and first solves the MDP for this case. The MDP is then used to generate a set of belief points for the PBVI solver. This is achieved by taking the Most Likely State (MLS) and to follow the MDP policy accordingly. It is orders of magnitude faster than HSVI and results in comparable polices. FSVI fairs badly however when information gathering actions are necessary. Since it is essentially using a myopic policy to generate its samples, these will be insufficient to find the global optimal policy when the solution requires information gathering actions. The very last sampling generation technique to date, which is considered to be the most efficient, is SARSOP (Kurniawati et al., 2008). It uses aspects from both HSVI and FSVI. It keeps upper and lower bounds on the value function and also uses the MDP solution to generate samples. The key idea of SARSOP is to sample belief points which will contain the optimal set of samples necessary to achieve an optimal policy. Both SARSOP and HSVI2 are considered state-of-the-art in PBVI value approximation techniques. See Du et al. (2010) for a review and comparison of both techniques on problems with thousands of states including simulation examples in grasping, tracking and UAV navigation. Other more recent approaches, Veiga et al. (2014), consider factorising the POMDP according to the different observations (independence assumption) and making use of linear function approximation methods to address the curse of dimensionality effecting the parameters of the POMDP (α-vectors).

These methods are well suited to addressing problems which are easily expressed in a discrete state space. All considered problems are simulation based and no physical interaction problems are considered. Besides the belief set generation problem, interest has also been poised on porting the PBVI to a continuous state space. An example of a continuous action space PBVI method is Perseus (Spaan and Vlassis, 2005), in which the authors replace the maximisation over the action by sampling the actions from a parametric continuous representation. In Porta et al. (2006) the state space, transition and observation model are represented by Gaussian Mixtures and the authors consider a particle set or Gaussian mixture representation of the belief. The authors show that a continuous representation of the state space preserves the PWLC property of the value function. They extend their method to continuous action and observations through sampling instead of discretising. Results are shown in a 1D continuous corridor setting. In a more recent approach Brechtel et al. (2013) a discrete state presentation of a continuous state space is learned and is combined with sampling techniques to solve the continuous integrals present in the Bellman equation. The explicit learning of the state representation leads to an increased performance when compared to the other continuous state PBVI methods.

PBVI techniques have come far since their first application to robotic navigation back in 2003 and have lead to a rapid increase of interest. Initially only a few hundred states could be considered and now problems with over tens of thousand of states are being solved in seconds (very problem specific of course). Most of the research has focused on how to gather a good set of sample beliefs efficiently. Later efforts focused on adapting PBVI to continuous state spaces more suited to robotic applications. The main approach consists of using sampling techniques to overcome the maximisation over the actions (when considering continuous actions) or to choose a suitable parametric representation of the transition, observation and utility model so that the Bellman equation can be solved in closed form. Most evaluations have focused on simulated and simplified robotic navigation problems in 1D and 2D. We have not discussed online POMDP-solvers since they are also based on VI and sampling techniques and thus share a lot of similarities with PBVI. We refer the reader to Ross et al. (2008) for a detailed review. In summary, trying to preserver the PWLC property of the value functions leads to complicated VI methods which are difficult to port to fully continuous state, action and observation space. Efforts which have attempted to do this have not yet been shown to scale. As a result of this difficulty, in making this transition to a fully continuous space, approximate value iteration methods have been explored as an alternative. In approximate value iteration the PWLC property of the value function is dropped and is represented by a regression function.

Point-based Value Iteration techniques try to preserve the PWLC property of the value function. This directly leads to a discretization of the state space which if continuous by nature, is prone to the curse of dimensionality. An alternative approach is to represent the value function by a non-parametric function, parameterize the belief space and perform approximate dynamic programming.

A very first successful example of this approach is Monte Carlo POMDP (MC-POMDP) Thrun (2000) in which a continuous state, action and observation version of the Heaven & Hell benchmark problem was solved successfully with a working implementation on a non-simulated mobile base. The belief was represented by a particle filter and the policy by a Q-value function, whose functional form was a non-parametric regressor (k-nearest neighbour) of the particle filter. The distance metric was the sample KL-divergence between two particle sets. The POMDP was solved through Reinforcement Learning (interaction with the environment) and approximated dynamic programming also known as experience replay, batch RL or Fitted Q-Iteration (FQI) Ernst et al. (2005b). Although computationally demanding the method was successful.

This inspired many similar approaches such as Brooks and Williams (2011) where the belief state filter was an Extended Kalman Filter (EKF), the value function was also non-parametric and the POMDP was solved via FQI. When compared with Perseus in a discretized 2D localisation task both approaches reached equivalent policies but the authors method achieved it far faster than Perseus, a PBVI method.

An alternative approach is to represent the history of the previous states or observations in an augmented state space and the treat the problem as a standard MDP. In this way the partial observability is directly encoded in the state representation. The motivation is that in contrast to POMDPs there has been far more research focused on MDPs and much work has been done on the application of non-linear function approximators for representing the value function in combination with reinforcement learning optimisation techniques to solving them. A successful example is the usage of a multi-layer perceptron as a Q-value function approximator, Neural Fitted Q-Iteration (NFQ) Riedmiller (2005). This approach was successfully applied to the standard RL benchmarking problems (carte pole, acrobat, mountain care), but no partially observable setting was considered. Later in Hausknecht and Stone (2015) the authors applied a Deep Recurrent Q-Network (DRQN) (extension to the work in Mnih (2015)) to capture the history of states in a game of Pong where the state space was occluded half the time. By introducing a long term memory component the POMDP in effect is turned into a MDP and the authors apply an optimisation approach similar to FQI.

The advantage of these approaches is that problem with very large state

spaces or continuous state spaces can be solved by using standard machine learning function approximation methods. These methods are easier to understand and implement and adapting POMDP methods to them is relatively straight forward. This is one particular way of dealing with the curse of dimensionality but not the only way. An alternative is to find a latent belief space which is of a much lower dimension than the original and perform value iteration in that space.

Latent Value Iteration

Latent belief space or belief space compression is a way of addressing the curse of dimensionality. The assumption is that although the belief space is of considerable size (thousands of dimensions) a latent belief space exists which is considerable smaller in terms of dimensions (a dozen). A first approach of compressing the belief is to transform it into a set of sufficient statistics (first and second moment for example) and treat the problem as a fully observable MDP in which the states are sufficient statistics of the beliefs. In Roy and Thrun (1999) the authors do just this, they compressed the filtered belief to its mean and entropy and performed VI on this augmented state space in a navigation task in which the goal was to reach a location with a minimum amount of uncertainty. This approach, called Augmented MDP (AMDP), brings a great simplification to solving the POMDP but at the cost of a lossy belief compression.

In further developments Roy (2005) compared both PCA and exponential-PCA (E-PCA) Roy and Gordon (2003), as a means of belief compression technique to find a low dimensional belief space. The authors showed that an original belief of thousands of dimensions could be compressed to a 10 dimensional belief space whilst retaining most of the information. This approach was shown to be superior to AMDP. It requires however computationally expensive transitions back and forth between the low and high dimensional belief states, a necessary step for the the application of VI. The latest work in this area is Li et al. (2010) which investigates the use of non-negative matrix factorisation in combination with k-means clustering as a way of compressing the belief. There method showed some improvement over the E-PCA approach but was only evaluated on discrete benchmark problems.

Belief compression as a means of reducing the curse of dimensionality is an interesting approach. The caveat is that it requires discretising the belief to a fixed grid, collecting many samples and learning an appropriate set of belief-basis eigenvectors. As such, the larger the state space, the larger the dimensionality and thus more samples are required to find a suitable set of basis belief-eigenvectors. Surprisingly, belief space compression methods have not had wide attention although they have shown promising results.

Value Iteration seeks to find an optimal policy directly through applying the Bellman equation to a belief-MDP (POMDP) and most of the research has focused on finding ways to alleviate the curse of dimensionality so that VI remains tractable in belief space. The first approach, PBVI, considers a relevant subset of the belief space. Because of the complexity involved in keeping the PWLC property of the value function which restricts its use in large state spaces, alternative approaches discard this property in favour of approximating the value function through machine learning regression techniques. These approaches are considerably more simple to implement than PBVI solvers which require heuristic pruning techniques and are difficult to port to continuous state spaces in general. Alternative approaches have considered finding a latent belief state and perform value iteration in this space. There has however been relatively little work in the latent belief space approach.

Overall, the above approaches consider mostly discrete actions even for the large state (history states) MDPs which have been gaining recent attraction. There are only a few exceptions and these resort to sampling strategies or the usage of paramaterized high level actions. The next approach we consider addresses the problem of continuous actions directly and are termed policy search methods.

2.3.2 Policy search

The approaches seen so far use a value function to encode the problem. When the optimal value function is solved, a policy can be derived from it by taking an action which maximises the value function at each time step, a process known as making the policy greedy with respect to the value function. This requires learning a high dimensional value function of the belief space and the resulting policies are not necessarily smooth, as small changes in the value function can lead to drastic changes in the policy. Even small approximation errors in the value function can lead to very bad greedy policies (Baxter and Bartlett, 2000). There is no doubt that deriving a policy from a generic value function for highly continuous policy, such as in the case of controlling an articulated robotic arm, is not easy.

This has lead to an alternate approach in which a policy is learned directly without a value function. An initial policy is defined in terms of a parameterized function, $\pi_{\boldsymbol{\theta}}$, and the utility is a function of the policy parameters, $u(\boldsymbol{\theta})$. The optimal policy is found by searching for the parameters $\boldsymbol{\theta}$ which will maximise the utility function. This is can be accomplished through various optimisation methods: gradient descent, gradient free, and expectation-maximisation.

Gradient: policy search

A very early type of policy search was REINFORCE (likelihood ratio) algorithms first introduced by Williams (1992). From a set of task executions, also called roll-outs, the gradient of the utility function is estimated and used to improve the policy through gradient ascent. The key aspect of this approach is that the derivative of the cost function is independent of the state transition model and as a result the gradient can be estimated by Monte Carlo methods. Application of this methodology to a partially observable setting lead to Gradient POMDP, GPOMDP (Baxter and Bartlett, 2000) in which the authors developed a conjugate stochastic gradient ascent algorithm to optimise a policy as a function of the current observations. To be optimal, the whole history should be considered or some sort of memory (compressed history) should be introduced. In an extension to this method Aberdeen and Baxter (2002), the authors used a HMM to parameterise models of the POMDP which they learned in conjunction with the parameters of the policy. These are early examples of policy search approaches which are able to fair well on the early POMDP benchmark problems such as "Heaven & Hell". The main difficulty is to reduce the bias and variance of the gradient estimate which preoccupies most gradient based approaches. Optimising the utility function via stochastic gradient ascent typically needs thousands of gradient estimates such that in expectation terms the parameters are maximising the cost function. An approach which mitigates this problem, coined Pegasus (Ng and Jordan, 2000), removes the stochasticity from the optimisation by setting the seed of the random number generator constant. A policy evaluation becomes deterministic and by repeating this process many times (different random seeds) the stochasticity is present between the different evaluations and not within them. The end result is the same as stochastic gradient ascent (if repeated sufficient times) but is far easier to optimise individual non-stochastic problems. This policy search method was used to learn a set of controllers for a radio controlled helicopter Kim et al. (2004), which is considered to be one of the very successful applications of RL to a MDP/POMDP problem. Recent approaches to gradient based methods include grasping objects under Gaussian position uncertainty Stulp et al. (2011), Stulp et al. (2012).

Expectation-Maximisation: policy search

One drawback of gradient based optimisation is that the learning rate plays a significant role on the speed of convergence. An alternative approach consists of using Expectation-Maximisation (EM) methods Kober and Peters (2009) which do not require a learning rate. Successful applications include: ball-in-a-cup, a humanoid learning the skill of archery Kormushev et al. (2010b), learning how to

flip a pancake Kormushev et al. (2010a) and keeping balance on a two-wheeled robot Wang et al. (2016). These are just some examples of the application of RL to continuous action and state space problems. When uncertainty is present, the maximum likelihood state estimate is typically taken and is treated as the true state. A good surveys on policy gradient search methods can be found in: Deisenroth et al. (2011), Kober et al. (2013).

Actor-critic: policy search

Gradient and EM methods only optimise the parameters of the policy, also known as actor only methods. An alternative is to have a separate parameterization of the value and policy functions. This approach is known as an **Actor-Critic**, in which the gradient of the utility function is used both to update the value and policy functions. It has been shown that this approach reduces the variance of the gradient estimate and allows to smoothly change the policy which is desirable when controlling a robot for instance, see Grondman et al. (2012) for a survey highlighting differences and advantages of policy search vs actor-critic methods. A successful application of actor-critic is (episodic) Natural Actor Critic (eNAC) Vijayakumar et al. (2003), Peters and Schaal (2008a), a method which uses the *natural gradient* of the value function to update the parameters of a policy. The advantage of using the natural gradient is that it guarantees small changes in the distance between the successive roll-out trajectory distributions. Previous policy gradient methods did not have such guarantees, since small parameter changes of the policy could lead to large changes in the roll-out distributions, which is undesirable. In terms of performance NAC converges faster than GPOMDP and has been applied to learn Dynamic Motor Primitives (DMPs) to control a humanoid robot.

Summary: policy search

For problems in which the state and action space are continuous, policy search is preferred to pure value iteration based methods, which is the case for articulated robotics. In this case, the policies are only guaranteed to be **locally optimal** as oppose to the VI methods which can find **global optimal** policies. However if the parameters of the policy are initialised such that it is in the vicinity of the global optimum of the utility function, then the local optimum will be global. A lot resides on the initialisation and dimensionality of the parameter space of the policy. In terms of using them to solve POMDPs, most examples, at least for robotic applications, act according to the Most Likely State (MLS) or are a function of a history of observations. In such a way the partial observability is **implicitly** encoded into the policy as opposed to explicitly as was the case for PBVI methods.

Policy gradient methods are iterative and generally require a lot of data to be able to achieve a good policy. Also often the policies learned are not transferable between different tasks and have to be completely relearned. This of course depends on the representation of the state space which if task invariant causes no problem, but unfortunately this is not the case. The next approach to treating uncertainty is more aligned with addressing this last issue of re-usability. These are the **planning** methods.

2.3.3 Planning

Belief space planners leverage the power of traditional planning and optimal control techniques such as: A-star (A*), Rapidly exploring Random Tree (RRT), Dijkstra and Linear-Quadratic Regulator (LQR) to the belief state space. In most of the following techniques (with a few exceptions), a fundamental assumption made is that the motion and measurement models are Gaussian and as a result, a point in the belief space can be represented by the first and second moment: the mean and covariance. An important distinction with VI and Policy search methods is that planners do not solve for a policy. These are online methods in the sense that they have to re-compute a set of actions at every time step as oppose to a policy which can directly query which action should be applied given the current state. The generic objective function used in belief space planning penalises for the amount of uncertainty at the goal and a cost is incurred for every step taken. The planned path will be a compromise between the exploitation actions, which seek to go directly to the goal, and information gathering actions, which seek to reduce the uncertainty.

Belief space road maps

An example of belief space planning is the application of Probabilist Road Maps (PMR) to a belief state space, Prentice and Roy (2009), referred to as Belief Road Maps (BRM). By taking advantage of the linear structure of the Kalman Filter update the authors show that the covariance matrix can be factorised such that a sequence of motion and measurement updates between two belief points in the BRM can be computed by a single linear operation parametrised by the current belief. The key advantage of this approach is that it allows for rapid replanning and is able to scale to large state spaces. The authors evaluated their planner in the MIT campus (simulated). Applications of this methodology include the control of an indoor quadrotor helicopter He et al. (2008) and indoor navigation (akbar Agha-mohammadi et al. (2011), akbar Agha-mohammadi et al. (2014)) (based on Feedback-based Information Road Maps FIRM , a similar approach in spirit to BRM).

Optimal control

Another main approach is based on optimal control theory, from which LQR have been adapted to a belief state space. In this setting the dynamics are considered linear (or linearizable) and the motion and measurement processes are Gaussian. The main difficulty of applying LQG to a belief space is that future observations are unknown, which implies that an expensive marginalisation of the observations has to be done. In Platt et al. (2010) the authors assume instead that at each time step the measurement obtained would be the **maximum-likelihood observation**. This assumption removes the stochasticity from the belief update (since the observations are considered known) and receding horizon optimisation techniques can be applied. These optimisation methods require a nominal trajectory which is generally generated assuming a fully observable state space with standard planning algorithms like RRT (Van Den Berg et al., 2011), and subsequently refined by dynamic programming methods until a local optimal solution in attained. In Erez and Smart (2010), the authors parametrized the belief by a mixture of two Gaussians to to tackle unilateral constraints and applied their planner to a 16 dimensional attention allocation problem. The optimisation method used was Differential Dynamic Programming (DDP) and maximum likelihood observations were assumed. For implementations based on this approach, when the planned belief trajectory deviates from the observed belief, replanning takes place. In recent improvements, van den Berg et al. (2012), the assumption of maximum-likelihood observation was removed successfully and has been applied in a simulated surgery problem, Sun and Alterovitz (2014), in which a needle has to be navigated through a body without entering into contact with vital organs.

Most optimal control methods assume that the belief space can be parametrized by a single Gaussian function, which can be restrictive. There have been a few approaches which consider **non-Gaussian** belief state spaces. In Platt et al. (2012) the authors introduced a non-Gaussian belief. The approach initially finds the Most Likely State (MLS) and then samples a set of hypothesis states from the belief. The cost function, with respect to the ML and sampled hypothesis, results in a sequence of actions which will seek to generate measurements which will prove or disprove the hypothetical states with respect to the ML state whilst also trying to reach the goal. Recent work Zito et al. (2013) incorporates this optimisation method into a grasping problem under non-Gaussian pose uncertainty. The method in question is able to perform well with only a few drawn samples from the belief. However the object was not picked up and as a result the stability of the grasp was not evaluated.

Summary: planning

Most advances in planning methods in belief space have been in optimal control

and were able to show applicability to high-dimensional belief state spaces in a variety of applications. To be fast, these methods have to make assumptions with respect to the shape of the belief (Gaussian) and the type of future observations which are available. These can be restrictive but in many applications (such as those which use vision) the uncertainty of objects in the world are often parametrized by Gaussians. The main difference between optimal control approaches and policy search methods is that the computational burden is shifted to online resolution of actions as opposed to constructing a policy offline through repeated interactions with the environment which can be very time consuming. The advantage of planning methods is that they are more flexible than parametric policies in the sense that they are more generic. They solve the objective function online and can be used in different environments, as oppose to a policy which would have to be re-learned.

2.3.4 Heuristics

The methods discussed so far can be considered computationally expensive and/or constraining in the type of belief which can be used (typically a unimodal Gaussian). If the problem domain is more complex or an expensive optimisation problem is not necessarily required, simple heuristic methods can achieve a satisfying solution and in some cases the equivalent of a full blown POMDP solver. Heuristic methods for dealing with uncertainty are widespread in robotics due to the high dimensionality and continuous state space. We consider here two heuristic approaches, myopic and information gain. Myopic ignores most of the variance in the uncertainty and considers only the Most Likely State (MLS) whilst information gain considers actions in terms of their uncertainty reduction.

Myopic & Q-MDP

Myopic policies consider only the most likely states, which in the case of a Gaussian belief is the mean, and act accordingly. These types of approaches ignore the variance in the uncertainty and risk to fail catastrophically or result in sub-optimal behaviour. MLS is typically used in complicated domains such as grasping, especially when the actual shape of the object is considered to be unknown. A successful approach to this problem is to have a prior non-parametric regressor function representing the shape. As contacts are made with the object more points are added to the regressor improving the shape constructed by exploring the unknown object and gradually acquiring points. The uncertainty of the shape in a region is typically a function of the number samples. At this point either an exploratory movement is done to move a finger towards a region of high uncertainty (the MLS region) or a grasping attempt

is carried out. In Hollinger et al. (2012) an AUV maps the hull of a ship by constructing a mesh and encoding the uncertainty of the mesh with a Gaussian Process (GP). A set of viewing locations, where there is uncertainty (MLS), are computed and a trajectory is obtained by solving a *travelling salesman* problem whilst seeking to maximise coverage of areas with high mesh uncertainty. In Chen and von Wichert (2015) a grasping controller uses the uncertainty, encoded by GP, to guide an exploration process. The fingers would move towards regions of high uncertainty whilst keeping contact with the object. For a good review on related methods for grasping objects under shape uncertainty consult Li et al. (2016), where the authors also use a GP based method to encode the shape uncertainty. The exploration methods for all these methods are in the same in spirit; move towards regions which have high uncertainty (exploration) and when the uncertainty is sufficiently low perform a grasp (exploitation).

An improvement is to consider the variance in the uncertainty and not just the MLS. Such an approach is a called Q-MDP Littman et al. (1995), Nowé et al. (2012) in which the underlying MDP is first solved assuming the state space to be fully observable. Then an action is taken which maximises the expected MDP value function weighted by the belief. This approach only considers uncertainty for one time step but it has been shown to be efficient in some domains (Thrun et al., 2005, Chap. 16). The negative aspect of this approach is that no information gathering actions emerge and the method will fail in problems where this is necessary (Heaven & Hell benchmark problem for instance). Most PBVI based research compare their algorithms against a Q-MDP agent and PBVI always fairs better. For a comparison of different heuristics such as Q-MDP and MLS consult Cassandra et al. (1996) and for a more recent comparison Lin et al. (2014). A recent application of this method include gaze allocation problems Nunez-Varela et al. (2012) where the uncertainty originates from the limited field of view. In Hauser (2011), Q-MDP is used to evaluate nominal trajectories generated from RRT where starting positions were sampled from the initial belief. A recent follow up on this idea, Vien and Toussaint (2015a), considers a task in which a robot has to localise itself with respect to a table. A set of macro actions are evaluated in a Q-MDP framework to achieve this task in which each macro action is solved by an optimal control method.

Both MLS and Q-MDP do not fully consider the uncertainty. This of course leads to great computational gain but at the expense of the quality of the policies, which can be very sub-optimal in some cases. It is known that for increasing the chance of success, a policy which deals with uncertainty needs both **goal orientated** and **information gathering** actions. The next heuristic approach, which we call **information gain**, is based on this concept.

Information gain

Information gain is the decrease or increase of uncertainty resulting from

the application of an action. It is obtained by computing the difference between the entropy of a forward simulated belief (through the virtual application of actions) and the entropy of the initial belief before the application of actions. The vast majority of applications consider a set of marco/parametrised actions. In this set there are typically goal orientation actions which will act as if the state space was fully observable (MLS move) and information gathering actions, whose goal is to reduce the amount of uncertainty such that the goal orientated actions have a higher chance to succeed. The cost function which is optimised is typically a compromise between the distance/time taken to reach the goal and the amount of information gained while executing the task. An early example considered path planning problem for a robot in the National museum of American history Roy et al. (1999). An information gain map was first computed off-line in which a map cell gave an estimate of how much information would be acquired at this location. This was incorporated into an objective function which optimised the information gain along a route with respect to the time taken to reach the goal. The path was given by solving the objective function using dynamic programming. In this case no explicit actions were defined, but the uncertainty was taken into account by weighting informative regions more than open space. The result was trajectories which stayed close to walls. Information gain methods are often used in SLAM applications because of the extremely high dimensionality of the belief space which is of the map and robot position. In Stachniss et al. (2005) a mobile robot is exploring and building a map of an office floor and a set of macro actions are available. A portion of the actions are exploratory and lead the robot to unexplored areas which results in an increase of uncertainty in the overall map whilst the other actions brings the robot back to already explored areas resulting in an improved estimate of the map. For each action, the information gain is computed and incorporated in an cost function. A one time step look head is done for each action, which potentially implies an expensive forward simulation, and the action giving the maximum information gain is chosen. This approach has been shown to be effective for large state space problems, notably in Active SLAM navigation Vallve and Andrade-Cetto (2014).

Information gain maximisation is not only restricted to navigation, there are many examples in grasping where this approach is used. Examples include tactile driven exploration such as in Hsiao et al. (2010) where a parametrised set of goal orientated and information gathering actions are used in the context of estimating the pose parameters (6D) of a power drill. The information gain of each action is incorporated into a cost function and the best action is chosen accordingly. The authors report a breadth first search depth of one action to achieve a good performance for the task. Later grasping approaches have built on this with different modifications to the information gain metric Javdani et al. (2013) and there have been successful applications such as finding a door handle Hebert et al. (2013) and opening a door.

Heuristic methods make strong assumptions which alleviates both the curse of dimensionality and the curse of history associated with POMDP problems. Either the MLS is considered (curse of dimensionality) or the planning horizon is restricted to one time step look ahead (curse of history) as it is the case for Information Gain methods. Heuristic methods in robotics have been regaining traction. In the early days of robotics methods such as Q-MDP, MLS, information gain maps and "best fields of view" were the predominant methods for considering uncertainty in policies and planning algorithms. This was simply due to the computational limitations of the time and POMDP solvers could only handle a few states before the arrival of PBVI methods. Since more sensory information is available and used in robotic systems it is again computationally expensive to compute optimal policies. In many cases spending large computational resources does not result in policies which are obviously superior to simple and intuitive heuristics. Lately many DARPA[2] teams when confronted with state uncertainty resort to information gain heuristics, for instance.

2.3.5 SUMMARY: LITERATURE

In the literature we characterised four approaches of how artificial agents have been programmed to reason under uncertainty.

When control algorithms were first being applied to mobile robotics uncertainty was handled with heuristics: MLS, Q-MDP and other techniques not fully discussed such as *next best view* methods. Practically speaking the computational resources at the time were too limited and it was practically infeasible to solve optimally for a POMDP problem. Also it is not clear at what time the robotic community started to apply results from operational research to robotics in the case of partial observability. Certainly it is not until the advent of the first point-based value iteration methods that there was a shift of interest towards improving POMDP solvers such that they could be applied to robotic domains (navigation & manipulation). When evaluated against heuristics methods it was clear that in some scenarios (Heaven & Hell problem) the POMDP solvers did far better. Value Iteration methods have not been widely used in cases where the action space is continuous. There have been efforts to adapt them to continuous actions space, however there is yet no concrete evidence that these methods scale. If the robotic domain requires continuous actions then either policy search or optimal control methods are preferred. Policy search methods were first considered since they are part of the Markov decision process family which is within the POMDP framework.

Policy search methods consider the uncertainty implicitly. These methods

[2] https://www.youtube.com/watch?v=9Oav3JajR7Q

work well when there is relatively few control parameters and behaviour to be learned are either reflexes (like in the case of the autonomous helicopter) or primitive actions such as picking up an object. The uncertainty considered in the reviewed literature on policy search methods is predominantly characterised by a Gaussian function. It is not clear how well policy search methods would scale to situations in which there is a lot of uncertainty and so far there has not been a lot of emphasis on comparing policy search methods with heuristics.

Optimal control methods came later, after policy search, and have recently started to gain traction since the adaptation of LQR to belief state spaces. As for policy search methods the uncertainty is considered Gaussian although recent research has been addressing this. The advantage of optimal control with respect to policy search methods is that they are more flexible since the objective function is resolved online. But at the expense of an increased computational cost.

Heuristics are still actively being used in research and very successful applications in the DARPA robotic challenge use heuristic approaches. The probable reason is the volume of sensory information and the size of the control architecture in robotic platforms competing does not leave room for anything else. Especially when considering project management constraints and reliability. That said, maybe there is no reason to use complicated methods. We note that there has been a significant absence of comparison between optimal control, policy search and heuristic problems on the same set of benchmark problems.

In Figure 2.8, we summarise attributes we consider important in the four approaches we reviewed. We bring attention to the typical type of actions and problems which these methods address. Note that we consider both Policy search and Value Iteration methods as being **off-line**. Although many authors say that the policy can be executed at any time, the optimal solution is not attained until after many interactions with the environment. This is not the case for Optimal Control and most Heuristic methods which give a solution on the spot, which we consider to be **on-line** methods.

The performance of all the methods mentioned in the literature review crucially depend on the quality of **exemplary demonstrations**. For instance, PBVI require search heuristics to find an optimal set of belief points, the quality of the optimal policy of policy search methods depend on the exploration-exploitation trade-off and optimal control methods strongly depend on the initial nominal trajectory. In a way this is intuitive, if you initialise your search method or algorithm with an initial solution which is of high quality (close to optimality) then which ever optimisation method used PBVI, Policy Search, Planning,... a solution should be obtained with computational ease. The question is then: *how to generate such exemplary demonstrations ?*

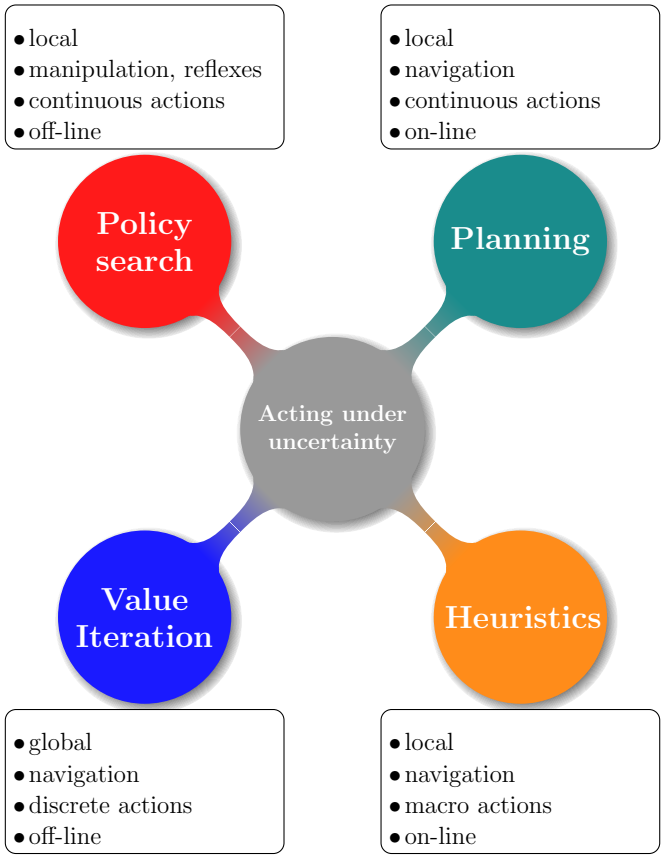


Figure 2.8: Summary of the aspects of the reviewed methods. Local refers to the optimality of the solution, on/off-line refers to if the solution is computed on the stop (on-line) or many simulations are required to obtain the solution (off-line).

2.4 Approach

As discussed in the literature summary, the initial data provided to the solvers plays an important role in the optimisation time and quality of the final policy or plan. A popular approach known as Programming by Demonstration (PbD) is a way to provide initial exemplary data. PbD is a methodology whose aim is to achieve the transfer of knowledge and behaviour from a teacher to an apprentice. The teacher is usually a human expert (this is not a constraint) who demonstrates to an apprentice how to accomplish a task. In the case of articulated robots, kinesthetic teaching is often preferred. The teacher would hold the robot, which is back drivable, and demonstrate to it trajectories. From the trajectories the states and actions, at each time step, are recorded and stored in dataset $\mathcal{D} = \{(x, a)\}$ which is then used to learn a a policy $\pi_{\boldsymbol{\theta}}(x, a)$, usually a regressor function, which encapsulates the taught behaviour. Other ways are possible such as using vision or a wearable interfaces which are common to both teacher and expert. We will not go into a great detailed review of PbD, for an in depth review the reader is referred to Billard et al. (2008), Billard and Grollman (2013). PbD has had many successful applications when the state space is considered observable but for the latent state case there are very few examples.

In this thesis we apply the PbD framework to a partially observable setting; we want humans to teach robots how to act under uncertainty. We know that generally speaking we are better at handling uncertainty than artificial agents, especially in haptic and tactile tasks. A hypothesis for this observation is probably that our perception capabilities are much higher and acute than current robotic software and hardware systems. To be able to study the ability of humans as teachers in a POMDP setting, we chose tasks in which a high level of uncertainty is present. For this reason we restrict ourselves to tasks in which the subjects can only use, their sense of touch. We namely consider **search tasks** in which a human is searching for an object whilst **blindfolded**. In summary we seek to learn control policies for robots in tasks which have the following problem specific attributes:

Problem attributes

- Blindfolded search tasks, no vision.
- Sparse haptic and tactile information.
- Continuous action and state space.
- High amount of non-Gaussian uncertainty.

In our approach, the robot apprentice observes the human teacher demon-

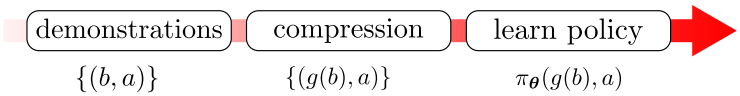


Figure 2.9: Three steps in learning a POMDP policy from human demonstrations: First gather the belief-action dataset, second compress the beliefs and third learn a generative policy.

strate a search task. As the human teacher searches, he makes contact with various aspects of the environment trying to localise himself whilst looking for the object in question. During the demonstration the apprentice infers the humans beliefs by observing his actions and stores them into a dataset $\mathcal{D} = \{(b, a)\}$. Given this belief-action dataset we learn a generative distribution $\pi_{\boldsymbol{\theta}}(b, a)$ of the behaviour exhibited during the search which is then transferred to the robot apprentice. The apprentice, we performing the search task, infers his own spatial beliefs and uses the learned generative model to extract appropriate actions, $a = \pi_{\boldsymbol{\theta}}(a|b)$. In Figure 2.9 we illustrate the PbD-POMDP data pipeline.

This is the general concept but there are a few caveats which make this task not as straight forward as it seems.

- **The belief state is unknown**: When the robot apprentice is watching the human perform a search task under state uncertainty, it is unable to observe the belief state of the human. All that the agent can observe directly are the actions of the teacher. We make **two assumptions**, the first is that the apprentice can infer the observations of the teacher by examining the teachers relation with the environment and secondly the initial uncertainty of the teacher is assumed to be known. From these two assumptions, the sequence of belief states can be inferred via a Bayesian filter. This implies that the mental belief state of the human teacher is in fact known given the assumptions. We give more details in Chapter 3 on the validity of these assumptions and discuss their relation to Bayesian Theory of Mind (BToM).

- **Learning a policy as a function of non-parametric beliefs:** Given that we are considering high levels of uncertainty and the observations are sparse, in the form of contacts, no parametrisation of the belief in terms of a Gaussian function would be adequate. In this thesis all the considered beliefs will be from the non-parametric Bayesian filter family, such as particle filters, which allows for a lot of flexibility. Learning a policy directly as a function of a particle filter is intractable. First in non-parametric filters there is typically thousands of states and in efficient particle filters the number of parameters varies over time. We **compress the belief** into the most likely state and the entropy. In this way the size of the belief state is fixed and low dimensional.

In Figure 2.10 we illustrate an example of human teaching an apprentice

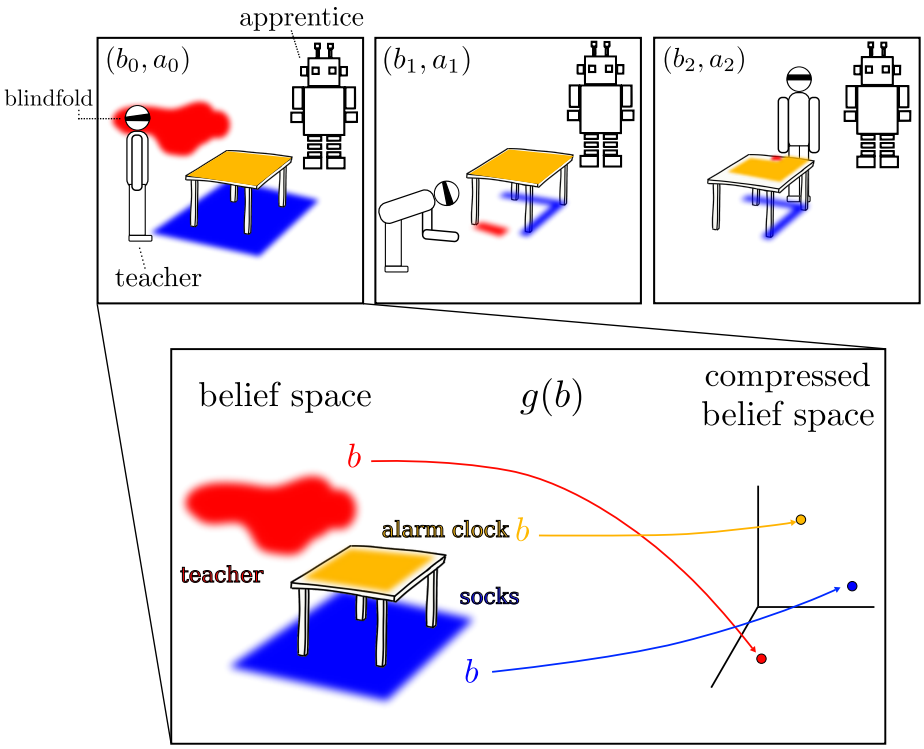


Figure 2.10: **Demonstrations:** An apprentice is looking at a human teacher who is searching for the alarm clock's button and his pair of socks. The apprentice assumes the structure of the original beliefs the human teacher has with respect to his position and that of the alarm clock and socks, these are represented by the red, yellow and blue density functions. **Compression:** Given the data set of beliefs and actions obtained from the demonstrations, the beliefs is compressed to a fixed parametrisation. **Learn policy:** A generative policy, $\pi_{\boldsymbol{\theta}}(g(b), a)$ is learned from the actions and compressed beliefs and can be executed according to the schematic on the right.

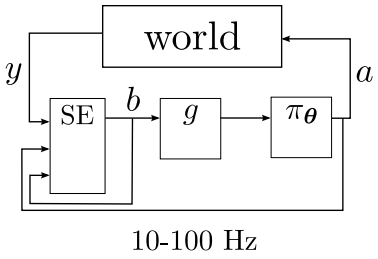


Figure 2.11: Control architecture of the apprentice robot. The control loop should run between 10-100Hz. Given an applied action, the world returns an observation which is integrated by the State Estimator (SE) to give the current belief. The belief is compressed and given as input to the policy.

robot how to search for objects (alarm clock and socks) in a state of high uncertainty, the human is blindfolded. Given what the apprentice can observe he must infer the beliefs of the teacher (red, blue and orange probability density functions).

- **Reactive policy**: The control loop cycle, which computes the belief state, compresses it, and computes the resulting action to take, should happen at around 10 to 100 Hz. This range may seem arbitrary but is in fact based on the humans control ability which at the highest cognitive level a delay in response is around 100ms and at the lowest reflex level at around 10-20 ms (Winter, 2009). This is to draw attention that the full control loop, belief filtering, compression and action prediction should all happen within this range. See Figure 2.11 for an illustration of the control architecture used.

- **Scalable belief filter**: In scenarios in which there are multiple objects being searched for by a human teacher, the joint belief distribution of a non-parametric Bayesian state space filter will become quickly computationally intractable. This motivates the development of a new type of SLAM filter methods which can scale in situations in which observations are very sparse.

All of the above points are the motivation behind many of the decision choices we take and use in the subsequent chapters. They are necessary such to be able to successfully teach robotic systems to act as humans in partial observable states.

Chapter 3

Learning to reason with uncertainty as humans

The conclusions drawn from the literature survey in Chapter 2 are that non-heuristic methods for planning and control rely heavily on the initial data provided to their respective optimisers. An ideal initial set of behaviour should be comprised of explorative and exploitative actions so that a final optimal policy can quickly achieve the balance between minimising uncertainty and solving the task at hand. This is especially true for Reinforcement Learning (RL) methods which make use of explorative actions to be able to find an optimal policy. In many RL applications random exploration or Gaussian noise perturbation is sufficient to find an optimal policy. This is the case when either an exhaustive search of the action space is possible (mountain cart, inverted pendulum, etc...) or in policy search methods where the policy is parametrised by a few parameters. In continuous action-state space POMDPs, when a generic non-parametric policy is desired this is not practical, especially when the decision horizon is long. Continuous action-state space POMDPs applications have predominantly focused on cases in which the uncertainty can be quantized by a single Gaussian parametrisation. This representation can be constraining since it requires the observation likelihood to be Gaussian as well. This assumption is restrictive and ill-suited for haptic search tasks in which observations are discontinuous and occur as impulses.

In this chapter, we demonstrate that human foresight and intuition can be leveraged as a means of solving the exploration/exploitation dilemma under partial observable conditions. Human beings are versatile in their ability to accomplish tasks which are considered to be complex by current robotic standards. This perceived ability which we have over current robotic systems, due to our prior domain knoweldge and experience, can be extracted, encapsulated and transferred to a robot apprentice. This chapther has been published in de Chambrier and Billard (2013, 2014).

To demonstrate the application of the transfer of behaviour from a human teacher to a robotic apprentice we apply the framework outlined in Chapter 2, Section 2.4 (PbD-POMDP) to a blindfolded haptic search task. In our blindfolded search task, both a robot and a human must search for an object on a table whilst deprived of vision and hearing, illustrated in Figure 3.1. The robot and human both have prior knowledge of the environmental setup making this

a specific search problem with no required mapping of the environment, also known as active localisation. In Figure 3.1, a human has his sense of vision and hearing impeded, making the perception of the environment partially observable and leaving only the sense of touch available for solving the task. The hearing sense is also impeded since it can facilitate localisation when no visual information is available and the robot has no equivalent giving an unfair advantage to the human. By impeding hearing we align the perception correspondence between the human and robot.

By representing the belief of the human's position in the environment by a Particle Filter (PF) and learning a mapping from this belief to hand actions (velocities) with a Gaussian Mixture Model (GMM), we can model the human's search process and reproduce it for any agent. We further categorize the type of behaviours demonstrated by humans as being either risk-prone or risk-averse and find that more than 70% of the human searches were considered to be risk-averse. We contrast the performance of this human-inspired search model with respect to Greedy and Coastal Navigation search methods. Our evaluation metric is the distance taken to reach the goal and how each method minimises the uncertainty. We further analyse the control policy of the Coastal Navigation and GMM search models and argue that taking uncertainty into account is more efficient with respect to distance travelled to reach the goal.

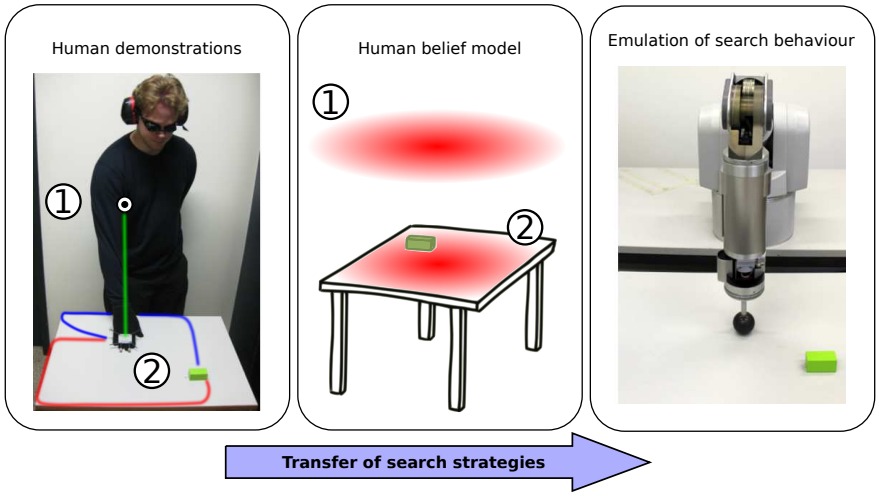


Figure 3.1: Blindfolded search task *Left:* Search task, a human demonstrator searching for the green wooden block on the table given that both his hearing and vision senses have been impeded. He starts (hand) at the white spot near position (1). The the red and blue trajectories are examples of possible searches. *Middle:* Inferred belief the human might have with respect to his position. If the human always starts at (1) and his belief is known, all following beliefs (2) can be inferred from Bayes rule. *Right:* WAM Robot 7 DOF reproduces the search strategies demonstrated by humans to find the object.

There are **two assumptions** we make when applying Programming by Demonstration, PbD (also known as Imitation Learning), to the POMDP task described above. The first assumption is that the human teacher's *spatial cog-*

nitive abilities are good enough to accomplish the task in a consistent fashion. In other words demonstrations should not be random and a pattern exists. The second assumption is that human beliefs inferred by the apprentice are close to the actual belief of the human.

3.1 Outline

- 3.2 Background
  We review aspects of the literature in robotics and cognitive science which are related to spatial navigation which consider scenarios with limited perceptual information. We review related literature from *Spatial Navigation*, *Theory of Mind* and *Programming by Demonstration*.

- 3.3 Experiment
  The table search experiment protocols are described and we detail how to learn and transfer search strategies from human teachers to a robot apprentice. A total of 15 human teachers participated and each gave 10 demonstrations, giving a total of 150 searches.

- 3.4 Formulation
  We detail the implementation of the human belief in terms of a Particle Filter (PF). This includes the measurement and motion models. We describe how we compress the belief particle filter in terms of the most likely state and differential entropy.

- 3.5 Policies
  - 3.5.1 Modelling human search strategies
    We detail the implementation and parametrisation of a Gaussian Mixture Model (GMM) policy encapsulating the human search strategies and how it synthesises new searches.
  - 3.5.2 Coastal Navigation
    We detail the implementation of a Coastal Navigation policy, used as a comparison with the GMM policy.

- 3.6 Results
  We conduct three types of analysis: we quantify the behaviour present in humans and policies in terms of riskiness; we qualitatively evaluate the differences between the GMM policy learned from human demonstrations and the Coastal Navigation policy; we evaluate the distance taken to find the goal for a set of four search policies, including the GMM.

3.2 Background

3.2.1 Spatial navigation

Spatial navigation, Wang (2007), Wolbers and Hegarty (2010), focuses on the role that sensory perception (vision, vestibular, proprioception ...), motor control and mental cognition have on the navigational ability of humans, animals and insects. A central aspect of spatial navigation is the way in which we mentally represent the geographical world, known as a *cognitive map* (mental representation of environment first proposed by Tolman, 1948) and how we update our pose estimation in this map. The aspects of both construction and correction of a cognitive map have been studied in great depth, Wolbers et al. (2008). There is reported evidence that we use both vestibular and proprioception in inferring self-motion in order to update our position through dead reckoning (also known as path integration). Given the estimated position we then use external cues such as geometric (the shape of a room) and features (the colour of the walls), to correct our position. The actual representation of our position and environment in our cognitive map has been proposed, Burgess (2006), to be either encoded in our own frame of reference (egocentric) or in a frame of reference which is independent to us (allocentric) and acts like a standard paper map or both. This cognitive map enables us to reason about the relations between our own position and that of other items and landmarks present. This representation also facilitates our ability to localise ourselves and plan novel routes when needed.

In Wang and Spelke (2000), the authors studied the effect that disorientation has on blindfolded subjects' ability to recover their heading, which is necessary for re-localisation. Through eight different experiments they concluded that humans have an egocentric cognitive map.

Studies have also looked at the difference between congenitally blind, late blind and sighted people in their ability to encode ego-allocentric cognitive maps. In Pasqualotto et al. (2013), the authors dispose a set of seven objects (brush, slipper, pan, dish, book, spoon, bottle) in the form of an array in a 12.5m $\times$ 9m room. The objects are positioned on top of stools. During a training phase, ten congenitally blind, ten late blind and ten blindfolded sighted people were taken through the setup and touched all objects present. This guided exploration (the experimenter leading the subject through the object array) was repeated until the participants could correctly recall all the objects' locations twice consecutively without help. In a testing room (no objects present) the participants were asked "Judgement of Relative Direction" questions and the accuracy and response time were recorded. From the results the authors concluded that blindfolded and late blinded participants used a allocentric representation of

the object array, whilst the congenital blind subjects use an egocentric model. The cause of this difference is attributed to the role played by vision in the development of the multisensory brain area, in which vision is necessary for the development of an allocentric model.

Many similar experiments have been conducted and a summary can be found in the following review Burgess (2006), where the authors explicitly state that a consensus has formed; both egocentric and allocentric representations of the environment are working in parallel. Current questions ponder whether allocentric models are part of the semantic memory as opposed to the procedural memory used by the egocentric model.

Spatial cognition and memory

The quality of the human teacher in search tasks, which are partially observable in the terms of absence of vision, will strongly depend on the teacher's ability to maintain an accurate cognitive map of his environment. This implies that the size of the environment and search task will have an effect on the teacher's ability to provide near optimal demonstrations. Early and influential research into human's short term memory was presented in 1956 by George Miller in a seminal work, Miller (1956) (22'780 citations), in which he described the "so called" magical number of our short term memory as being 7 ± 2 items, known as *Miller's Law*. This research was conducted on a one dimensional task in which no spatial navigation was required. Since then there have been many studies investigating the limits of short term memory.

In Lavenexa et al. (2015) a set of subjects had to find either 1, 3, 5 or 7 goal pads, among a grid array of 23 pads in a 4m $\times$ 4m room, within a one minute interval. They measured the subjects' error in terms of the number of locations visited before finding the goals. They found that on average the subjects had to visit "$1.6 \times \#num\_goals$" pads before achieving the task. The authors concluded that in this spatial navigation task there was no magical number which represents the limit of short term memory. In another spatial navigation experiment, Iachini et al. (2014), the effect that the scale of the environment has on the ego-allocentric representation is studied in blindfolded, late and early blind subjects. The main findings were that cognitively blind people have more difficulty in developing an allocentric representation of the world.

In Stankiewicz et al. (2006), a search task in a virtual maze is conducted by a set of human subjects. The aim is to investigate the limitations that perception, memory and uncertainty have on human decisions in comparison to an ideal agent (POMDP solution). The authors' main findings were that as the size of the maze increases the performance of the human subject decreases with respect to the ideal agent, as human subjects are limited by the uncertainty in their location and have difficulty in maintaining multiple hypotheses.

The studies detailed above reported that if the environment is not overly large and complex our cognitive model is sufficient to produce policies which are on par with an optimal POMDP agent.

Our study seeks to transfer exploratory behaviour from human teachers to a robot apprentice in a partially observable setting. In our search scenario the environment is less than 3 meters in length and 2 meters in depth with a single goal object to be found. Given this setup and the evidence from previous studies, humans should be able to achieve this task with a high level of proficiency.

This is beneficial since currently both humans and animals are better at spatial navigation than robots Stankiewicz et al. (2006) especially when uncertainty is present. The quality of the demonstrations will strongly depend on the teacher's short term memory in retaining a sufficiently accurate cognitive map of the environment.

3.2.2 Human beliefs

A crucial aspect for the success of PbD-POMDP learning is that the apprentice be able to infer the human's belief of his location whilst he is searching. In others words the apprentice (human or robotic) has to infer the cognitive map of the teacher.

The study of inference of another's mental state is part of Theory of Mind (ToM) Sodian and Kristen (2010), which is concerned with our ability to infer beliefs, desires, intentions, perception, goals and current knowledge. In this study, the apprentice will have to infer the teacher's beliefs which we assume are **rational**. A rational belief is a belief for which observations bring supportive evidence and gradually increase the certainty of the belief. In a recursive formulation this known as Bayesian Theory of Mind (BToM), where the Bayesian component highlights the hypothesis that humans integrate information and update their beliefs in a similar fashion to Bayes rule.

Due to the complexity in the number of sensory sub-components, such as gaze following, and their interplay, required as a precursor to the development of a ToM, much effort has been focused their development. Early work in implementing a ToM in a humanoid robot was introduced in Scassellati (2002) and is based on ToM models of Leslie (1994) and Baron-Cohen (1995). The author focused on building basic skills such as face finding and distinguishing animate and inanimate stimuli but left open the problem of the final interaction between all the components.

In Butterfield et al. (2009), the authors model ToM as a Markov Random Field which defines a joint probability distribution over a set of hidden actions and observation variables. The functions of these variables are hand-crafted for

each experiment. The authors demonstrate that the MRF achieved inference capabilities close to those of ToM. Recently in Devin and Alami (2016), ToM and planning architecture have been integrated in a joint action collaborative human robot task, in which position, goal and action state of the human partner is maintained by his robotic assistant.

Work on modelling human beliefs and intentions has been undertaken in cognitive science, Baker et al. (2011), Richardson et al. (2012). In Baker et al. (2006), the authors present a Bayesian framework for modelling the way humans reason and predict actions of an intentional agent. The comparison between a generic Bayesian model and the humans' predictions yielded similar inference capabilities. This provided evidence supporting the hypothesis that humans integrate information using Bayes rule. Further, in Baker et al. (2011), a similar experiment was performed in which the inference capabilities of humans, with regard to both belief and desire of an agent, were compared to that of their Bayesian model. Again they found that human's inference was comparable to that of the Bayesian model.

In our PbD-POMDP framework we make a similar hypothesis that humans integrate information in a Bayesian way, however in a continuous domain. We infer the belief that humans have of their location in the world during search tasks.

3.2.3 Programming by demonstration & uncertainty

Programming by demonstration (PbD) is advantageous in the POMDP and MDP contexts since it removes the need to perform the time consuming exploration of the state-action tree to discover an optimal policy and does not rely on any exploration heuristics to gather a sufficient set of belief points (as in point based value iteration methods discussed in Chapter 2).

We expect humans to perform an informed search. In contrast to stochastic sampling methods, humans utilise past experience to evaluate the costs of their actions in the future and to guide their search. This foresight and experience are implicitly encoded in the parameters of the model we learn from the demonstrated searches.

PbD has a long history in the autonomous navigation community. In Kasper et al. (2001), behaviour primitives of the PHOENIX robot control architecture are incrementally learned from demonstrations. Two types of behaviour namely *reactive* and *history-dependent* are learned and are encoded by radial basis functions. The uncertainty is implicitly handled by directly learning the mapping between stimulus and response. In Hamner et al. (2006) the parameters of a controller which performs obstacle avoidance are learned from human demonstrations. The uncertainty is inherently handled by learning the relation between sensor input and control output. In Silver et al. (2010) the objective function of

a path planner is learned from human demonstrations. The objective function is a weighted sum of features corresponding to raw sensor measurements. This is another example where the partial information of the state is taken into account at the perception-action level, with the difference that instead of a policy being learned the objective function from which it is generated is learned.

Uncertainty is not restricted to state estimations but can also present in dynamical interaction with the environment in which unforeseen and perceptual uncertainties can arise such as in manipulation tasks. When solely tracking a Cartesian trajectory position uncertainties (poor visual estimation of the target) can lead to the failure of the task or a dangerous accumulation of contact forces. In Pastor et al. (2011) the authors learn, via imitation learning, an initial Dynamic Movement Primitive (DMP) Cartesian policy and separately a target a force profile. By using sensor feedback they modify the target trajectory so as to replicate the force profile thus achieving robustness to pose uncertainty. Another possibility is to vary the stiffness parameters of an impedance controller Kronander and Billard (2012) based on the position uncertainty, if the position uncertainty is high then the robot will be more compliant. In Medina et al. (2013) the authors introduce a risk-sensitive control framework which depending on the uncertainty will make a trade-off between the position and force error tracking gains of an impedance controller. The task (Cartesian position and velocity) which is tracked by the controller is learned from demonstrations and encoded in a model.

Much work has been undertaken in learning reactive-behaviour, history dependent behaviour and combining multiple behaviour primitives to achieve complex behaviour. However very few have studied the effect of uncertainty in the decision process and do not consider it during the learning or assume that it is implicitly handled. A noticeable exception is Lidoris (2011), in which a human expert guides the exploration of a robot in an indoor environment. The high level actions (*Explore*, *Loop Closure*, *Reach goal*) taken by the human are recorded along with three different features related to the uncertainty in the map. Using SVM classification a model is learned which indicates which type of action to take given a particular set of features. The difference with our approach is that we perform the learning in continuous action space at trajectory level and multiple actions are possible given the same state, which cannot be handled by a classifier.

3.3 Experiment: table search

In our search task setup, Figure 3.2 and Figure 3.3 (*top left*), a group of 15 human volunteers were asked to search for a wooden green block located at a fixed position on a bare table. Each participant repeated the experiment 10 times from four starting points with an associated small variance. The starting

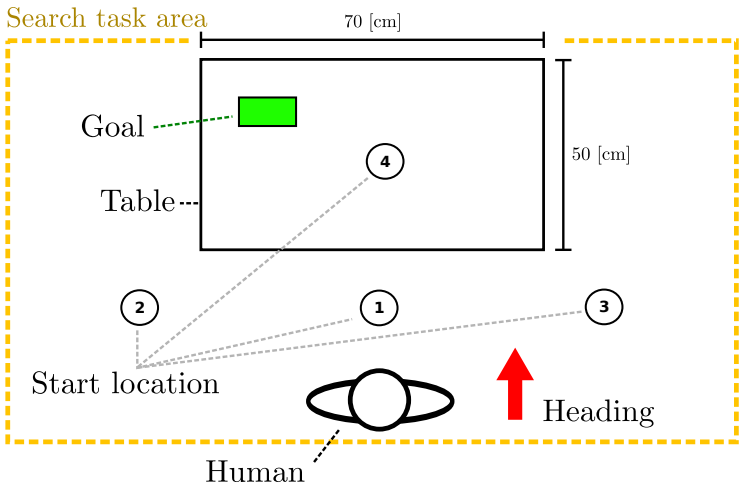


Figure 3.2: Table search task. Blindfolded human subjects after a disorientation step are placed in one of the four starting locations. The heading of the subject is always kept the same. The human's objective is to locate the green block on the table. Throughout all experiments the green wooden block is kept in the same location.

positions were given with respect to the location of the human's hand (all participants where right handed). The humans were always facing the table with their right arm stretched out in front of them. The position of their hand was then either in front, to the left, to the right, or in contact with the table itself.

As covered in the background section, previous work has taken a probabilistic Bayesian approach to model the beliefs and intent of humans. A key finding was that humans update their beliefs using Bayes rule (shown so far in the discrete case). We make a similar assumption and represent the human's location belief (where he thinks he is) by a particle filter which is a point mass representation of a probability density function. There is no way of knowing the human's belief with certainty. We make the critical assumption that the belief is observable in the first time step of the search and all following beliefs are assumed correct through applying Bayes integration. The belief is always initialized to be uniformly distributed on top of the table, see Figure 3.3 (*top right*), and the starting position of the human's hand is always in this area.

Before each trial the participant was told that he/she would always be facing the same direction with respect to the table (so always facing the goal, like in the case of a door) but his/her translational starting position would vary. For instance, the table might not be always directly in front of the person and his/her distance to the edge or corner could be varied. In Figure 3.3 *bottom left*, we illustrate four representative recorded searches whilst in the *bottom right*, we illustrate a set of trajectories which all started from the same region. One interesting aspect is the diversity present, demonstrating clearly that humans behave differently given the same situation.

It is non-trivial to have a robot learn the behaviour exhibited by humans performing this task. As we cannot encapsulate the true complexity of hu-

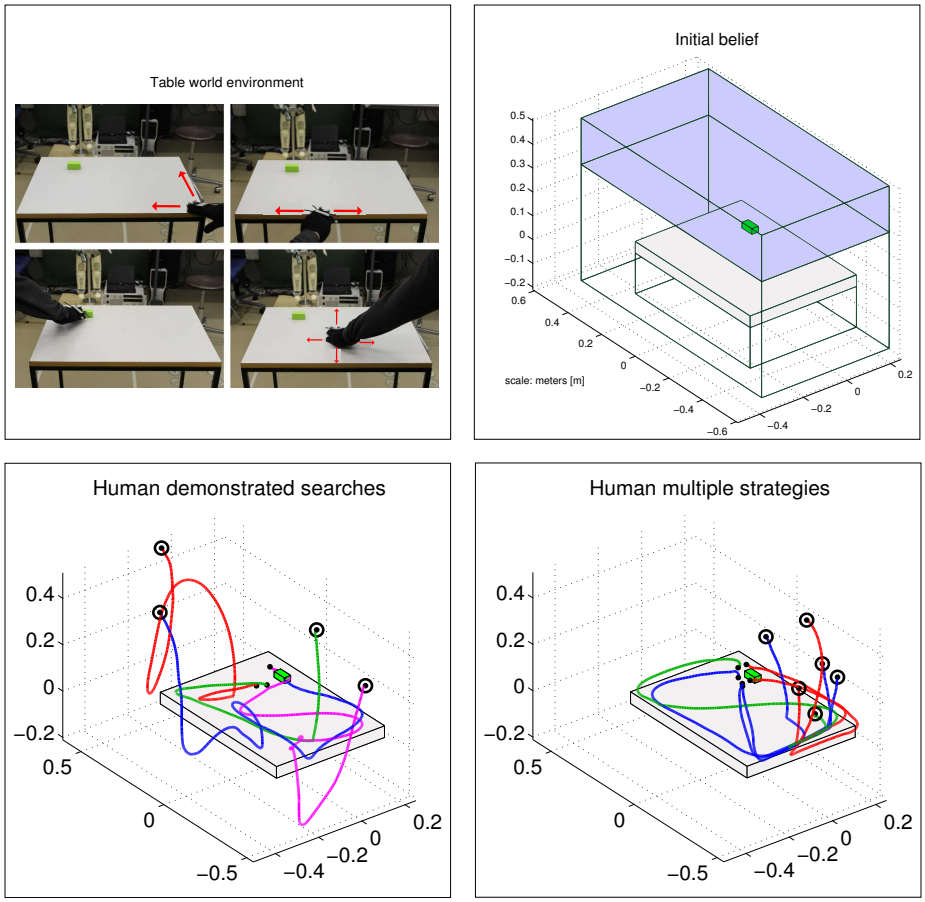


Figure 3.3: *Top left*: A participant is trying to locate the green wooden block on the table given that both vision and hearing senses have been inhibited. The location of his hand is being tracked by the OptiTrack® system. *Top right:* Initial distribution of the uncertainty or belief we assume the human has with respect to his position. *Bottom left:* Set of recorded searches, the trajectories are with respect to the hand. *Bottom right:* Trajectories starting from same area but have different search patterns, the red trajectories all navigate to the goal via the top right corner as opposed to the blue which go by the bottom left and right corner. Among these two groups there are trajectories which seem to minimize the distance taken to reach the goal as opposed to some which seek to stay close to the edge and corners.

man thinking, we model the human's state through two variables, namely, the human's uncertainty about his current location and the human's belief of his position. The various strategies adopted by humans are modelled by building a mapping from the state variables to actions, which are the motion of the human arm. Aside from the problem of correctly approximating the belief and its evolution over time, the model needs to take into consideration that people behave very differently given the same situation. As a result it is not just a single strategy that will be transferred but rather a mixture of strategies.

3.4 Formulation

In the standard PbD formulation of this problem, a parametrised function is learned, mapping from state x_t, which denotes the current position of the demonstrator's hand, to the hand's displacement $\dot{x}_t$. In our case since the environment is partially observable we have a belief or probability density function, $p(x_t|y_{0:t}, \dot{x}_{1:t})$, which is conditioned on all sensing information, $y_{0:t}$, (the subscript, $0:t$, indicates the time slice which ranges from, $t = 0$, to the current time, $t = t$) over the state space at any given point in time and the history of applied actions, $\dot{x}_{1:t}$. We seek to learn this mapping, $f : p(x_t|y_{0:t}, \dot{x}_{1:t}) \mapsto \dot{x}_{t+1}$, from demonstrations. During each demonstration we record the following variables:

- $\dot{x} \in \mathbb{R}^3$, normalised Cartesian velocity of the hand.
- $\hat{x} = \arg\max_{x_t} p(x_t|y_{0:t}, \dot{x}_{1:t})$, end-effector's most likely position.
- $U \in \mathbb{R}$, entropy $H\left(p(x_t|y_{0:t}, \dot{x}_{1:t})\right)$.

We define the belief state to be the compressed vector $b \in \mathbb{R}^4$,

$$b = \begin{bmatrix} \hat{x} \\ U \end{bmatrix} \tag{3.4.1}$$

A statistical controller was learned from a dataset of N demonstrations: $\mathcal{D} = \{(\dot{x}^{[i]}_{1:T}, b^{[i]}_{1:T})\}_{i=1:N}$, where the upper index $[i]$ referenced the ith search trajectory, recorded during the search trials of the human subjects. Having described the experiment we proceed to give an in-depth description of the mathematical representation of the belief, sensing and motion models and the uncertainty.

Belief model

A human's belief of his location in an environment can be multi-modal or uni-modal, Gaussian or non-Gaussian and may change from one distribution to

another. We chose a particle filter to be able to represent such a wide range of probability distributions. A particle filter is a Bayesian probabilistic method which recursively integrates motion and sensing to estimate a posterior from a prior probability density. The particle filter is comprised of two models. The first, the *motion model* $p(x_t|x_{t-1}, \dot{x}_t)$ estimates a distribution over the next possible states. The second, *sensing model* $p(y_t|x_t)$, corrects the motion updated distribution through integrating sensing information. These two models are recursively applied and achieve a Bayesian filter in which a prior distribution over the state space is turned into a posterior distribution. The two steps are formalised below.

$$p(x_t|y_{0:t-1}, \dot{x}_{1:t}) = \int p(x_t|x_{t-1}, \dot{x}_t)\, p(x_{t-1}|y_{0:t-1}, \dot{x}_{1:t-1})\, dx_{t-1} \quad (3.4.2)$$

$$p(x_t|y_{0:t}, \dot{x}_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{0:t-1}, \dot{x}_{1:t})}{p(y_t|y_{0:t-1})} \quad (3.4.3)$$

The probability distribution over the state $p(x_t|y_{0:t}, \dot{x}_{1:t})$ is represented by a set of weighted particles which represent hypothetical locations of the end-effector and their density which is proportional to the likelihood. The particular particle filter used was the *Regularised Sequential Importance Sampling* (M. Sanjeev et al., 2002, p.182). From previous literature Baker et al. (2011) it has been shown that there is a similarity between Bayes update rule and the way humans integrate information over time. Under this assumption we hypothesise that if the initial belief of the human is known then the successive update steps of the particle filter should correspond to a good approximation of the next beliefs.

Sensing model

The sensing model gives the likelihood, $p(y_t|x_t)$, of a particular sensation y_t given a position $x_t \in \mathbb{R}^3$. In a human's case, the sensation of a curvature indicates the likelihood of being near an edge or a corner. However the likelihood cannot be modelled using the human's sensing information. Direct access to pressure, temperature and such salient information is not available. Real sensory information needs to be matched against virtual sensation at each hypothetical location x_t of a particle. Additionally, for the transfer of behaviour from human to robot to be successful, the robot should be able to perceive the same information as the human, given the same situation. An approximation of what a human or robot senses can be inferred, based on the end-effector's distance to particular features in the environment. In our case four main features are present, namely corners, edges, surfaces and an additional dummy feature defining no contact, air. The choice of these features is prior knowledge given to our system and not extracted through statistical analysis of recorded trajec-

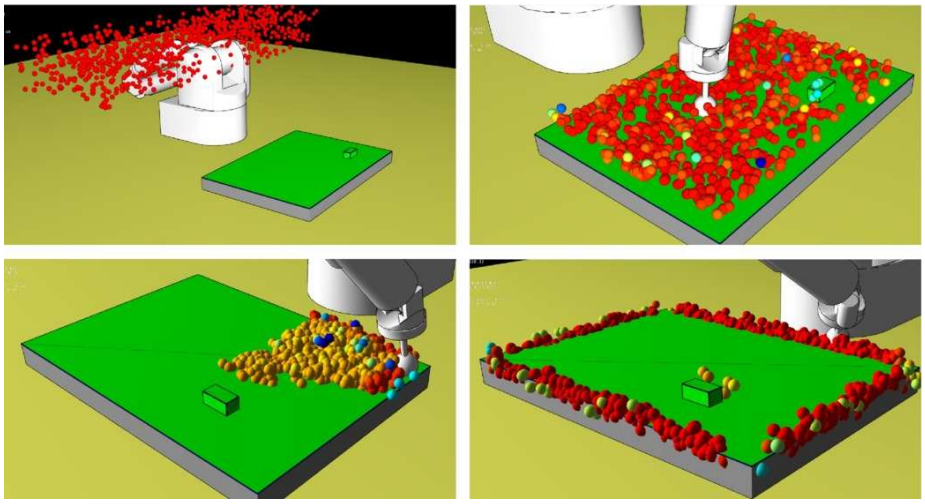


Figure 3.4: Four different time frames of the evolution of the belief particle filter. *Top left*: Initial belief distribution; a lot of uncertainty. *Top right:* First contact is made with the table, the measurement likelihood restrains the samples to be on the table's surface. *Bottom right:* First contact is an edge. *Bottom left:* Gradual localisation.

tories. The sensing vector is $y_t = [p_c, p_e, p_s, p_a]^{\mathrm{T}}$, is a categorical distribution and prior to normalisation p is the probability of a feature being sensed and the subscript corresponds to the first letter of the feature it is associated with. In Equation 3.4.4, the sensing function, $h(x_t, x_c)$, returns the probability of sensing a corner, where $x_c \in \mathbb{R}^3$ is the Cartesian position of the corner which is the closest to x_t.

$$p_c = h(x_t, x_c; \beta) = \exp\left(-\left(\beta \cdot \|x_t - x_c\|\right)^2\right) \tag{3.4.4}$$

The exponential form of the function, h, allows the range of the sensor to be reduced. We set $\beta > 0$ such that any feature which is more than 1cm way from the end effector or hand has a probability close to zero of being sensed. The same sensing function is repeated for all feature types.

The sensing model takes into account the inherent uncertainty of the sensing function 3.4.4, and gives the likelihood, $p(y_t|x_t)$ of a position. Since the range of sensing is extremely small and entries are probabilistic we assume no noise in the sensor measurement. The likelihood of a hypothetical location x_t is proportional to the Jensen-Shannon divergence (JSD) $p(y_t|x_t) = 1 - JSD(y_t||\hat{y}_t)$ between true sensing vector y_t obtained by the agent and that of the hypothetical sensation $\hat{y}_t$ generated at the location of a particle. In Figure 3.4, four different beliefs are shown.

Motion model

The motion model is straight forward compared with the sensing model.

In the robot's case the Jacobian gives the next Cartesian position given the current joint angles and angular velocity of the robot's joints. From this the motion model is given by $p(x_t|x_{t-1}, \dot{x}_t) = \mathrm{J}(q)\dot{q} + \epsilon$ where q is the angular position of the robot's joints, $J(q)$ is the Jacobian and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ is white noise. The robot's motion is very precise and its noise variance is very low. For humans, the motion model is the velocity of the hand movement provided by the tracking system. In our experiment we consider the noise from motion to be negligible. An increase in uncertainty already results from the re-sampling stage of Sampling Importance Resampling (SIR) particle filter and we found no need to add additional motion noise. The particles' positions were updated by applying the measured velocity obtained from either the visual tracking system (when recording the human demonstrations) or the robot's forward kinematics.

Uncertainty

In a probability distribution framework, entropy is used to represent uncertainty. It is the expectation of a random variable's total amount of unpredictability. The higher the entropy the more the uncertainty, likewise the lower the entropy, the less the uncertainty. In our context, a set of weighted samples replaces the true probability density function of the belief, $p_{\boldsymbol{\theta}}(x_t|y_{0:t}, \dot{x}_{1:t})$. A reconstruction of the underlying probability density is achieved by fitting a Gaussian Mixture Model (GMM), Equation 3.4.5, to the particles,

$$p_{\boldsymbol{\theta}_H}(x_t|y_{0:t}, \dot{x}_{1:t}) = \sum_{k=1}^{K} w^{[k]}\, g(x_t\,; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}) \tag{3.4.5}$$

where parameters $\boldsymbol{\theta}_H = \{w^{[k]}, \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{1:K}$, are the weights, means and covariances of the individual multivariate Gaussian function, $g(\cdot)$ and K is the number of Gaussian components. The scalar $w^{[k]}$ represents the weight associated to mixture component k (indicating the component's overall contribution to the distribution) and $\sum_{k=1}^{K} w^{[k]} = 1$. The parameters $\boldsymbol{\mu}^{[k]} \in \mathbb{R}^{(3\times 1)}$ and $\boldsymbol{\Sigma}^{[k]} \in \mathbb{R}^{(3\times 3)}$ are the mean and covariance of the normal distribution k.

The main difficulty here is determining the number of parameters of the density function in a computationally efficient manner. We approach this problem by finding all the modes in the particle set via mean-shift hill climbing and set these as the means of the Gaussian functions. Their covariances are determined by maximizing the likelihood of the density function via Expectation-Maximization (EM).

Given the estimated density we can compute the upper bound of the differential entropy $H(\cdot)$ (Huber et al., 2008),

$$H\left(p_{\boldsymbol{\theta}_H}(x_t|y_{0:t}, \dot{x}_{1:t})\right) = \sum_{k=1}^{K} w^{[k]} \left(-\log(w^{[k]}) + \frac{1}{2}\log((2\pi e)^D|\Sigma_k|)\right) \tag{3.4.6}$$

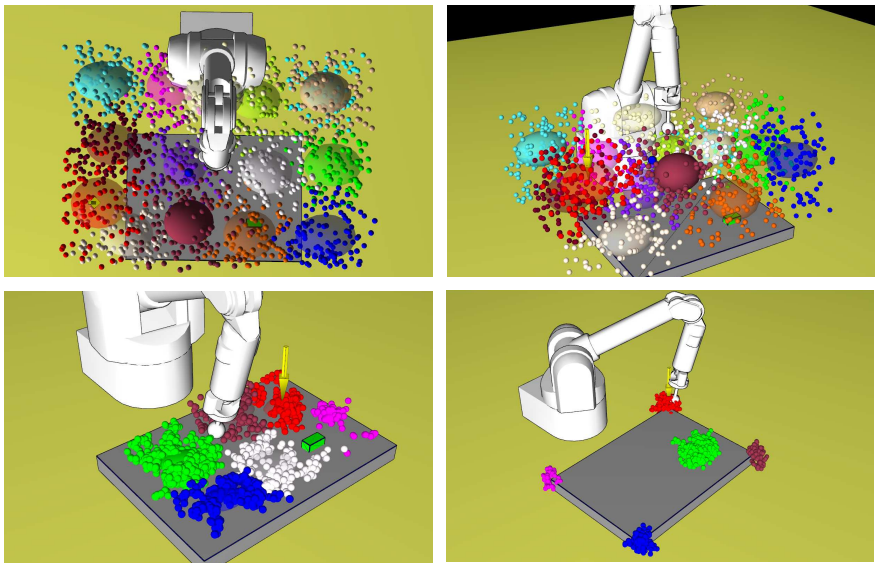


Figure 3.5: Representation of the estimated density function $p_{\boldsymbol{\theta}_H}(x_t|y_{0:t}, \dot{x}_{1:t})$. *Top Left and Right:* Initial starting point, all Gaussian functions are uniformly distributed with uniform priors. The red cluster always has the highest likelihood which is taken to be the believed location of the robot's/human's end-effector. *Bottom Left:* Contact with the table has been established, the robot location differers from his belief. *Bottom Right:* Contact has been made with a corner, the clusters reflect that the robot could be at any corner (note that weights are not depicted, only cluster assignment).

where e is the base of the natural logarithm and D the dimension (being 3 in our case).

The reason for using the upper bound is that the exact differential entropy of a mixture of Gaussian functions has no analytical solution. When computing both the upper and lower bounds it was found that the difference between the two was insignificant, making any bound a good approximation of the true entropy. The choice of the believed location of the robot/human end-effector is taken to be the mean of the Gaussian function with the highest weighted w.

$$\hat{x}_t = \arg\max_{x_t} p_{\boldsymbol{\theta}_H}(x_t|y_{0:t}, \dot{x}_{1:t}) = \boldsymbol{\mu}^{[\arg\max_k(w^{[k]})]} \tag{3.4.7}$$

Figure 3.5 depicts different configurations of the modes (clusters) and believed position of the end-effector (indicated by a yellow arrow).

3.5 Policies

3.5.1 Modelling human search strategies

During the experiments, the recorded trajectories show that different actions are present for the same belief and uncertainty making the data multi-modal (for

a particular position and uncertainty different velocities are present). That is multiple actions are possible given a specific belief. This results in a one-to-many mapping which is not a valid function, eliminating any regression technique which directly learns a non-linear function. To accommodate this fact we use a GMM to model the human's demonstrated searches. Using statistical models to encode control policies in robotics is quite common, see Billard et al. (2008).

By normalising the velocity the amount of information to be learned was reduced. We also took into consideration that velocity is more specific to embodiment capabilities: the robot might not be able to reproduce safely some of the velocity profiles demonstrated.

The training data set comprised a total of 20'000 tuples $(\dot{x}, b)$, from the 150 trajectories gathered from the demonstrators. The learned GMM $\pi_{\boldsymbol{\theta}}(\dot{x}, b)$ had a total of 7 dimensions, $\dot{x} \in \mathbb{R}^3$ and $b \in \mathbb{R}^4$. The definition of the GMM is presented below in Equation 3.5.1.

$$\pi_{\boldsymbol{\theta}}(\dot{x}, b) = \sum_{k=1}^{K} w^{[k]}\, g(\dot{x}, b; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}) \tag{3.5.1}$$

The parameters $\boldsymbol{\theta} = \{w^{[k]}, \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{1:K}$, are the weights, means and covariances of the individual Gaussian functions, $g(.)$,

$$\boldsymbol{\mu}^{[k]} = \begin{bmatrix} \boldsymbol{\mu}_{\dot{x}}^{[k]} \\ \boldsymbol{\mu}_{b}^{[k]} \end{bmatrix}, \; \boldsymbol{\Sigma}^{[k]} = \begin{bmatrix} \boldsymbol{\Sigma}_{\dot{x}\dot{x}}^{[k]} & \boldsymbol{\Sigma}_{\dot{x}b}^{[k]} \\ \boldsymbol{\Sigma}_{b\dot{x}}^{[k]} & \boldsymbol{\Sigma}_{bb}^{[k]} \end{bmatrix}$$

where $\sum_k w^{[k]} = 1$, $\boldsymbol{\mu}_{\dot{x}}^{[k]} \in \mathbb{R}^3$ and $\boldsymbol{\mu}_{b}^{[k]} \in \mathbb{R}^4$. Given this generative representation of the humans' demonstrated searches we proceeded to select the necessary parameters to correctly represent the data. This step is know as model selection and we used Bayesian Information Criterion (BIC) to evaluate each set of parameters which were optimised via Expectation-Maximisation (EM).

A total of 82 Gaussian functions were used in the final model, 67 for trajectories on the table and 15 for those in the air. In Figure 3.6 *(left)* we illustrate the model learned from human demonstrations where we plot the 3 dimensional slice (the position) of the 7 dimensional GMM to give a sense of the size of the model.

3.5.2 Coastal Navigation

Coastal navigation (Roy et al., 1999) is a path planning method in which the objective function, Equation 3.5.2, is composed of two terms.

$$f(x_{0:T}) = \sum_{t=0}^{T} \lambda_1\, c(x_t) + \lambda_2\, I(x_t) \tag{3.5.2}$$

The first term $c(x_t)$ is the traditional "cost to go" which penalizes every step taken so as to ensure that the optimal path is the shortest. The value was

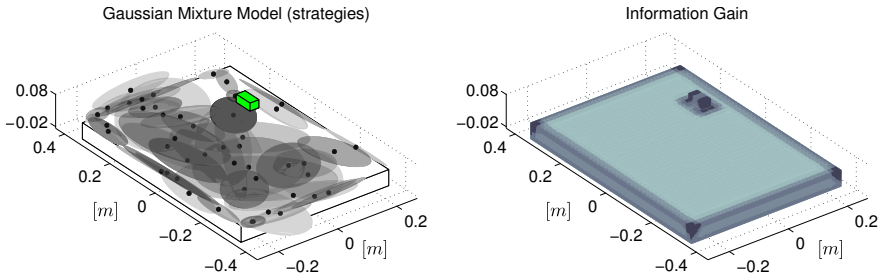


Figure 3.6: *Left:* Resulting search GMM, a total of 67 Gaussian mixture components are present. We note the many overlapping Gaussians: this results from the level of uncertainty over the different choices taken. For example humans follow along the edge of the table in different directions and might leave the edge once they are confident with respect to their location. *Right:* Information Gain map ($I(x)$ Equation 3.5.3) of the table environment, dark regions indicate high information gain as oppose to lighter ones. Not surprisingly, the highest are the corners, followed by the edges.

simply set to 1 for all discrete states in our case. The second term, $I(x_t)$, is the information gain of a state. The information gain of a particular state is related to how much the entropy of a probability density function (pdf), being the location's uncertainty in our case, can be reduced. The two λ's are scalars which weigh the influence of each term. There is no constraint that they must sum to one.

In our table environment we discretised the state space, $\mathbb{R}^3$, into bins so as to have a resolution of approximately, $1\,cm^3$, giving us a total of a 125'000 states. The action space was discretised to 6 actions, two for each dimension meaning that all motion is parallel to the axis. For each state x an $I(x)$ value is computed by evaluating Equation 3.5.3,

$$I(x) = \mathbb{E}_{p(y_t|x_t)}\{H(p_{\boldsymbol{\theta}_H}(x_t|y_{0:t}, \dot{x}_{1:t})\} - H(p_{\boldsymbol{\theta}_H}(x_t|y_{0:t-1}, \dot{x}_{1:t})) \tag{3.5.3}$$

which is essentially the difference between the entropy of a prior pdf to that of a posterior pdf. We set our initial pdf to be uniformly distributed and we computed the maximum likelihood sensation for each discrete state x_t which is akin to the expected sensation or assuming that there is no uncertainty in sensor measurement (an assumption often made throughout the literature to avoid carrying out the integral of the expectation in Equation 3.5.3). The result is the difference between the posterior pdf, given that the sensation occurred in x_t, and the prior pdf. The resulting cost map is illustrated in Figure 3.6. As expected, corners have the highest information gain followed by edges and surfaces. We do not show the values above the table since they provided much less information gain.

The optimization of the objective function is accomplished by running the Dijkstra's algorithm. This algorithm, given a cost map, computes the shortest path to a specific target from all the states. This results in a policy.

3.5.3 Control

The standard approach to control with a GMM is to condition on the belief state b and perform inference on the resulting conditional GMM, Equation 3.5.4, which is a distribution over velocities or directions.

$$\pi_{\boldsymbol{\theta}}(\dot{x}|b) = \sum_{k=1}^{K} w_{\dot{x}|b}^{[k]} \; g\left(\dot{x}\,; \boldsymbol{\mu}_{\dot{x}|b}^{[k]}, \boldsymbol{\Sigma}_{\dot{x}|b}^{[k]}\right) \tag{3.5.4}$$

The new distribution is of the dimension of the output variable, the velocity (3 dimensions). The variable $\dot{x}$ in $\dot{x}|b$ indicates the predictor variable and the variables b has been conditioned. A common approach in statistical PbD methods using GMMs is to take the expectation of the conditional (known as Gaussian Mixture Regression), Equation 3.5.5

$$\dot{x} = \mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\} = \sum_{k=1}^{K} w_{\dot{x}|b}^{[k]} \; \boldsymbol{\mu}_{\dot{x}|b}^{[k]} \tag{3.5.5}$$

The problem with this expectation approach, is that it averages out opposing directions or strategies and may result in a net velocity of zero. One possibility would be to sample from the conditional, however this can lead to non-smooth behaviour and flipping back and forth between modes resulting in no displacement. To maintain consistency between the choices and avoid random switching we perform a weighted expectation on the means so that directions (modes) similar to the current direction of the end-effector receive a higher weight than opposing directions. For every mixture component k, a weight α_k is computed based on the distance between the current direction and itself. If the current direction agrees with the mode then the weight remains unchanged but if it is in disagreement a lower weight is calculated according to the equation below.

$$\alpha_k(\dot{x}) = w_{\dot{x}|b}^{[k]} \; \exp(-\cos^{-1}(< \dot{x}, \boldsymbol{\mu}_{\dot{x}|b}^{[k]} >)) \tag{3.5.6}$$

Gaussian Mixture Regression is then performed with the normalised weights α instead of wi (the initial weight obtained when conditioning).

$$\underline{\dot{x}} = \mathbb{E}_{\alpha}\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\} = \sum_{k=1}^{K} \alpha_k(\dot{x}) \; \boldsymbol{\mu}_{\dot{x}|b}^{[k]} \tag{3.5.7}$$

The final output of Equation 3.5.7 gives the desired direction ($\dot{x}$ is re-normalised and is denoted as $\underline{\dot{x}}$). In the case when the mode suddenly disappears (because of sudden change of the level of uncertainty caused by the appearance or disappearance of a feature) another present mode is selected at random. For example, when the robot has reached a corner, the level of uncertainty for this feature drops to zero. A new mode, and hence new direction of motion, will then be computed. However this is not enough to be able to safely control the robot. One needs to control the amplitude of the velocity and ensure compliant con-

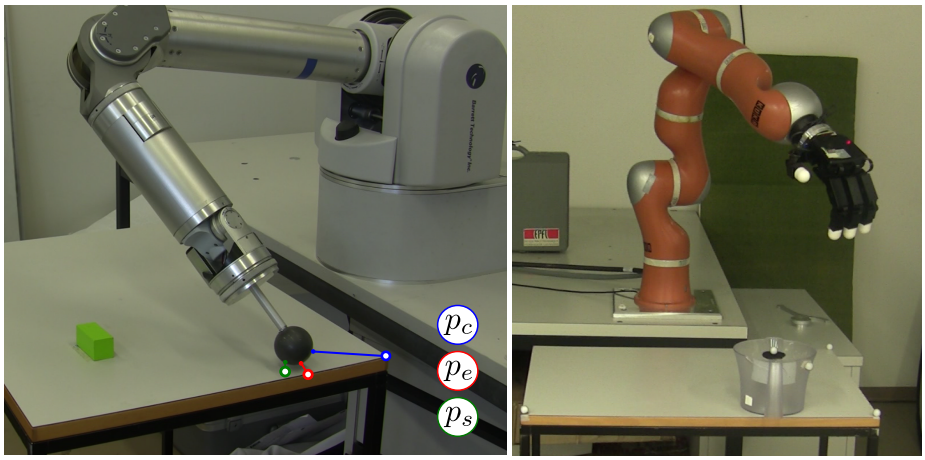


Figure 3.7: *Left:* The WAM is cable driven 7 Degrees of Freedom (DOF) robot and is controlled by directly sending torque commands. *Right*: The KUKA LWR is as the WAM a 7 (DoF) robot. Both robots are controlled via an Ethernet cable at a 1kHz communication rate. The KUKA API provides a command interface to the stiffness, damping, position and torque variables of each joint in contrast to the WAM for which our own impedance controller was developed. In our setup the end-effector of the WAM is a haptic device from which we computed sensations y, by treating it as a point mass with respect to the model of the world and use Equation 3.4.4 on page 59 to compute the measurements. The KUKA is equipped with the Allegro hand from which we compute sensations from each finger tip, also by considering geometric distances with features in the environment.

trol of the end-effector when in contact with the table. This behaviour is not learned here, as this is specific to the embodiment of the robot and unrelated to the search strategy. The amplitude of the velocity is computed by a proportional controller based on the believed distance to the goal,

$$\begin{aligned} \nu &= \max(\min(\beta_1, K_p(x_g - \hat{x}), \beta_2) \\ \dot{x} &= \nu \, \underline{\dot{x}} \end{aligned} \tag{3.5.8}$$

where the β's are lower and upper amplitude limits, x_g is the position of the goal, and K_p the proportional gain which was tuned through trials.

3.5.4 Robot implementation

The above procedure can control the general behaviour of the search but is insufficient for a successful implementation on a robotic system such as the 7 Degree of Freedom $q \in \mathbb{R}^7$ (q is a vector of joint positions) WAM or KUKA LWR robot, which we illustrate in Figure 3.7. The GMM policy $\underline{\dot{x}} = \mathbb{E}_\alpha\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\}$ outputs a linear velocity and the angular velocity is computed from a reference orientation which is constant. From both linear and angular velocities a reference position $x^r \in \mathbb{R}^{(3\times 1)}$ and orientation $R^r \in \mathbb{R}^{(3\times 3)}$ are computed and used to define a linear and angular error $x_e = x^r - x$, $\psi_e = \text{angleaxis}(R^{\mathrm{T}} R^r)$ by using the the current position x and orientation R of the robot's end-effector. Given

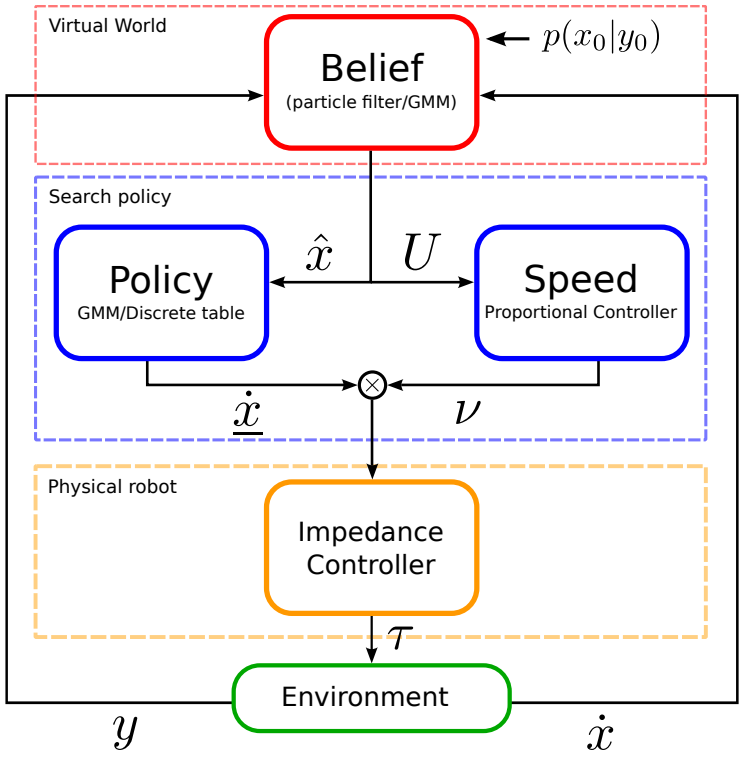


Figure 3.8: Overview of the decision loop. At the top a strategy is chosen given an initial belief $p(x_0|y_0)$ of the location of the end-effector (initially through sampling the conditional). A speed is applied to the given direction based on the believed distance to the goal. This velocity is passed onwards to a low level impedance controller which sends out the required torques. The resulting sensation, encoded through the Multinomial distribution over the environment features, and actual displacement are sent back to update the belief.

the kinematic chain of the robot, the inverse of the Jacobian $J(q) \in \mathbb{R}^{6\times 7}$ is used in an impedance control to transform the Cartesian error $c_e = [x_e, \psi_e]^{\mathrm{T}} \in \mathbb{R}^{6\times 1}$ to torque commands $\tau_t \in \mathbb{R}^7$, Equation 3.5.9,

$$\tau_t = J^{\mathrm{T}}(q_t)\left(-Kc_e - D\dot{c}_e\right) + g(q_t) \tag{3.5.9}$$

where $K, D \in \mathbb{R}^{6\times 6}$ are diagonal stiffness and damping matrices whose values were set experimentally and $g(q_t)$ compensates for gravity. Given an applied torque there is a resulting joint velocity $\dot{q}_t$ from which we can compute the measured Cartesian end-effector velocity used in the motion model of the particle filter. The learned search strategies were evaluated both on the WAM and KUKA illustrated in Figure 3.7. Figure 3.8 illustrates the complete control flow.

3.6 Results and discussion

Throughout our evaluation of our GMM PbD-POMDP control policy we

will be considering four search policies: Greedy, GMM, Hybrid and Coastal. We evaluate behaviour present in the human demonstrations, and the four above mentioned policies in terms of their risk. We qualitatively compare the policies of the GMM model and the Coastal Navigation algorithm and highlight the effect of uncertainty. We finish with a quantitative evaluation of search efficiency in terms of distance travelled until the goal is found. The outline of this section follows as:

- Section 3.6.1, we analyse the types of behaviour present in the human demonstration as well as in four different search algorithms: Greedy, GMM, Hybrid and Coastal.
- Section 3.6.2, we qualitatively analyse the GMM search policy (namely the different modes/decisions present) with respect to the Coastal navigation policy.
- Section 3.6.3, we evaluate the search performance, with respect to the distance taken to reach the goal and the uncertainty profiles towards the end of the searches in 5 different experiments (different types of initializations).

3.6.1 Search & behaviour analysis

For each method (Greedy, GMM, Hybrid, Coastal) 70 searches were performed with all starting positions drawn from the uniform distribution used during the teaching stage (depicted in Figure 3.3 *top right*, page 56). In Figure 3.9 we illustrate the expected sensation $\mathbb{E}\{y\}$ and variance $\text{Var}\{y\}$ for each trajectory with respect to the edge and corner of the table.

The selection of edges and corners as features as a means of classifying the type of behaviours present is not solely restricted to our search task. Salient landmarks will result in a high level of information gain, which is the case for the edge and corner (see Figure 3.6 *right*, page 63). Other tasks can use such features or variants in which the curvature is considered for representing the task space. These features are present in most settings and high level features can use these easily as their building blocks.

We note that the Greedy search approach seeks to go directly to the goal without taking into account the uncertainty. The GMM models human search strategies. The Hybrid is a combination of both the Greedy and GMM method where once the uncertainty has been sufficiently minimised, the policy switches (threshold) to the Greedy method for the rest of the search. The Coastal navigation algorithm finds the optimal path to the goal based on an objective function which consists of a trade-off between time taken to reach the goal and the minimisation of the uncertainty.

It can be seen that the human demonstrations have a much wider spread than those of the search algorithms. We speculate that this is due to human

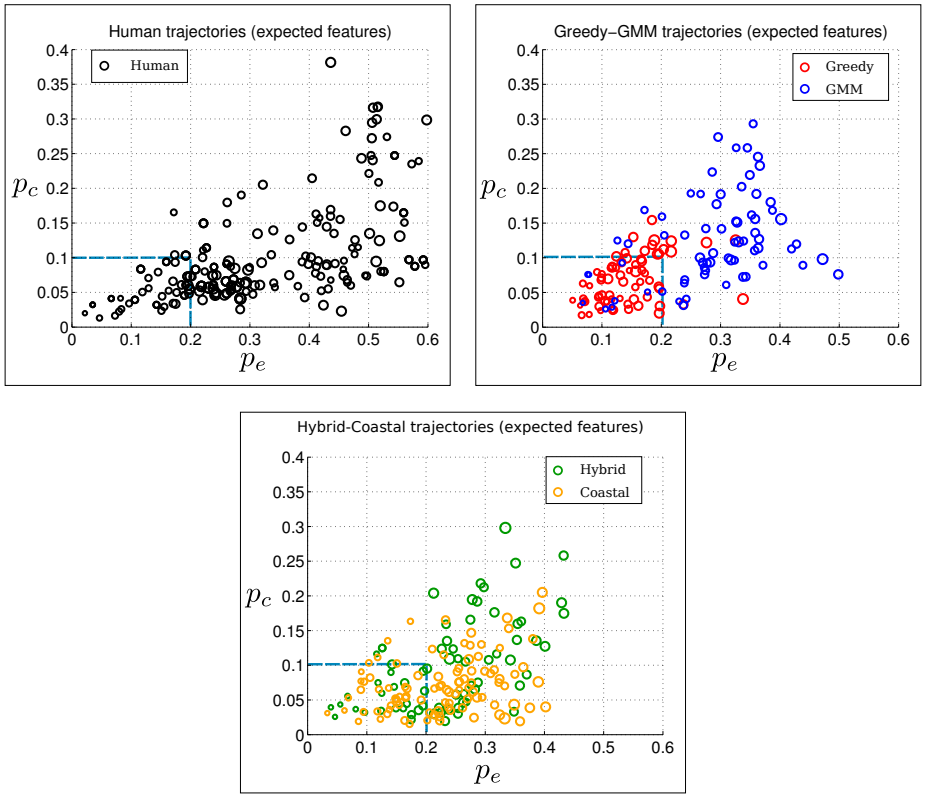


Figure 3.9: Expected sensation. Plots of the expected sensation of the edge and corner feature for all trajectories. The axes are associated with the sensor measurements, 0 means that the corresponding feature is not sensed and 1 the feature is fully sensed. A point in the plots summarises a whole trajectory by the mean and variance of the probability of sensing a corner or edge. The radius of the circles are proportional to the variance. The doted blue rectangle represents the decision boundary for classifying a trajectory as being either risk-prone or risk-averse. A point which lies inside the rectangle is risk-prone. *Left:* Human trajectories demonstrate a wide variety of behaviours ranging from those remaining close to features to those preferring more risk. *Right:* Red points show Greedy and blue points the GMM model. *Bottom:* Green circles are associated with the Hybrid method whilst orange are those of the Coastal navigation method. The Hybrid method is a skewed version of the GMM which tends towards risky behaviour and exhibits the same kind of behaviour as the Coastal algorithm.

Criteria	**Greedy**	**GMM**	**Hybrid**	**Coastal**	**Human**
risk-prone (f)	77 %	11 %	30 %	46 %	26 %
risk-prone (r)	78 %	12 %	24 %	45 %	7 %

Table 3.1: Percentage of risk-prone trajectories based on two decision criteria, the feature (f) and the risk (r) (information gain) metrics discussed above.

behaviours being optimal with respect to their own criteria as opposed to the algorithms which usually tend to only maximise a single objective function. The trajectories of the Greedy and GMM methods represented by their expected features demonstrate two distinctive behaviours (in terms of expected sensation), risk-prone for the Greedy and risk-adverse for the GMM.

We make **the assumption** that Greedy trajectories are risk-prone by nature. We performed a SVM classification on the Greedy-GMM expected features (Figure 3.9 *right*) and used the result to construct a decision boundary as a means of classifying a trajectory as being either risk-prone or risk-averse. Table 3.1 *first row* shows that the GMM and Human search trajectories are mostly risk-averse. Surprisingly the Coastal policy seems to be very risk-prone given that it seeks paths close to highly informative areas. We use a second metric based on the information gain, which we call the Risk factor, to classify trajectories as being either risk-prone or risk-averse.

The Risk factor of each individual trajectory is inversely proportional to its accumulated information gain. Figure 3.10 (*left*) shows the kernel density estimation distribution of the risk for each search method. Two trajectories per search type corresponding to a supposed risk-prone and risk-averse search are plotted in the expected feature space in Figure 3.10 (*right*). As expected, risk-prone strategies for which the risk tends to 1 have a low expectation of sensing edges and corners and produce trajectories with a low information gain while those with a high expectation of sensing features have a high information gain. Since the metric lies exclusively in the range [0,1] we define that every trajectory which has a Risk factor lower than than 0.5 will be considered risk-averse whilst those above are risk-prone. Table 3.1 *second row* illustrates the riskiness of each search method. It is evident that humans are risk-averse in general followed by GMM which is a smoothing of the human data, then Hybrid which as expected should be more risk-prone since it is a linear interpolation between the GMM and Greedy search policies and finally Coastal and Greedy.

Figure 3.11 (*top left & right*), shows risk-prone (red) and averse (green) trajectories produced by human demonstrations and by the Greedy search. Both these extremes correspond to our intuition that risk-averse trajectories tend to remain closer to features or areas of high information gain as oppose to risk-prone searches. However to stress the case that humans have multiple search strategies present, we performed 40 GMM searches (model of the human behaviour) which all started under the same initial conditions (same belief distribution, true position and believed position). Figure 3.11 shows the resulting trajecto-

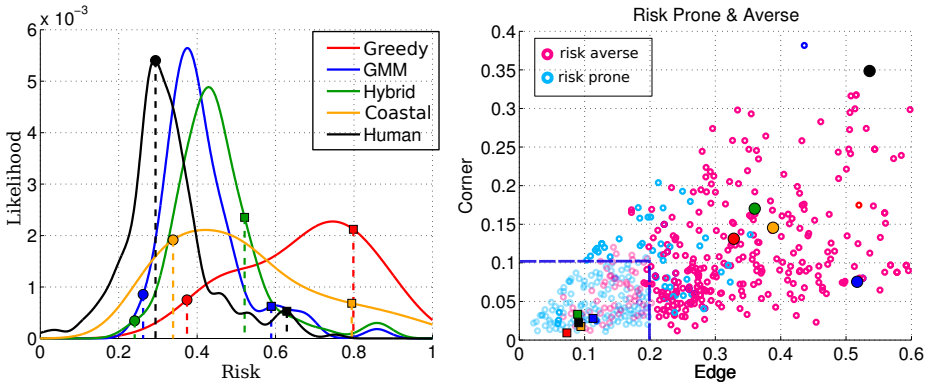


Figure 3.10: Risk of searches. Illustration of risk-prone and risk-averse searches in terms of a Risk factor (*left*) and expected sensation (*right*). *Left:* Each trajectory was reduced to a single scalar, which we call the Risk factor, quantifying the risk of a trajectory. The Risk factor is inversely proportional to the sum of the information gain of a particular trajectory. The colour paired dots (risk averse) and squares (risk prone) represent trajectories which are plotted in Figure 3.11, to illustrate that these correspond to risk averse and prone searches. *Right:* Corresponding trajectories chosen in the Risk factor space but represented in the feature space. As expected, trajectories with a high risk map to regions of low expected feature. However the transition from the Risk space to feature space is non-linear and will result in a different risk-level classification than the feature metric previously discussed.

ries and expected features for each trajectory. It is clear that multiple searches occur which is reflected in the plot of the expected features. All of the search strategies generated by the GMM for this initial condition produced risk-averse trajectories.

We conclude that there is evidence of multiple search strategies present in the human searches since they were extracted and encoded in the GMM model. From the risk distribution, humans have a tendency to be risk-averse.

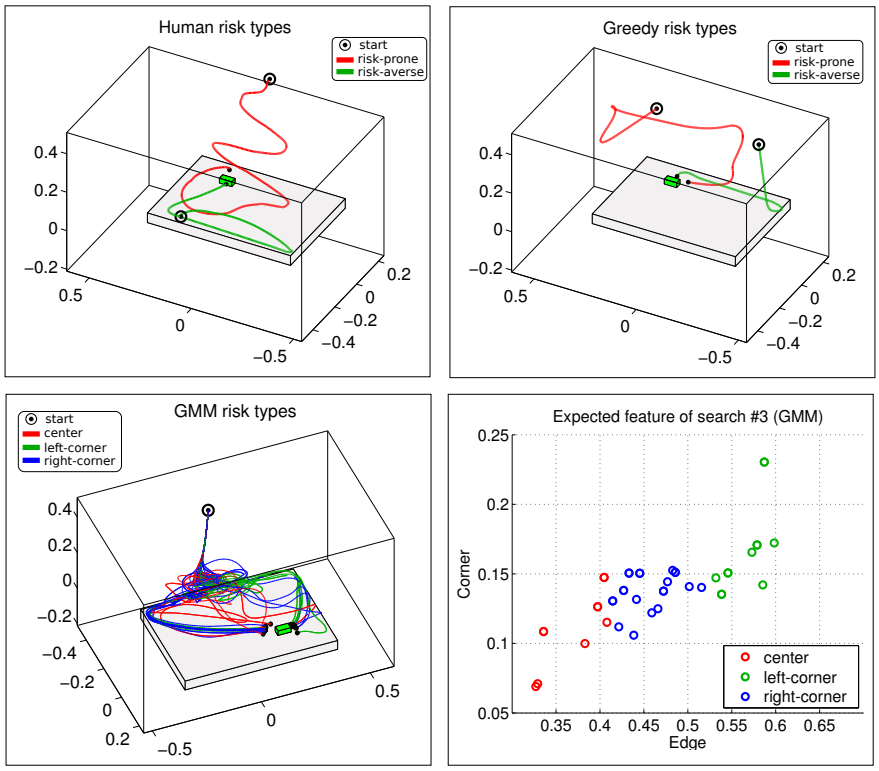


Figure 3.11: Risk prone & averse searches (red & green trajectories). *Top left:* Two human trajectories taken from data shown in Figure 3.10. *Top right:* Two Greedy trajectories. *Bottom left:* GMM trajectories, all starting from the same location, the colour coding is to illustrate the different policies which were encoded and emerge given the same initial conditions. *Bottom right:* Corresponding expected features of each trajectory, the colour coding matches the trajectories to the "GMM risk types" sub-figure. All the searches which were generated by the GMM for this initialisation produced risk-averse searches (based on the feature metric discussed previous).

3.6.2 GMM & Coastal Navigation policy analysis

We next illustrate some of the modes (action choices) present during simulation and evaluate their plausibility. Figure 3.12 shows that multiple decision points have been correctly embedded in the GMM model. All arrows (red) indicate directions that reduce the level of uncertainty.

Figure 3.13 depicts the vector fields of both Coastal and GMM models where, as expected, the Coastal navigation trajectories tend to stay close to edges and corners until they are sufficiently close to the goal. This is achieved by weighting the information gain term $I(x_t)$ in the objective function sufficiently (λ_2). If λ_2=0 the Coastal policy is the same Greedy algorithm.

It can be further seen that when the uncertainty tends towards it's maximum value ($U \rightarrow 1$) all behaviour tends to go towards the edges and corners. As the uncertainty reduces ($U \rightarrow 0$) the vector field tends directly towards the goal. However even at a low level of uncertainty, the behaviour at the edges and corners remains multi-modal and tends to favour remaining close to the edges and corners. This is an advantage of the GMM model. If the uncertainty has been sufficiently reduced and the true position of the end-effector or hand is not near an edge the policy dictates to go straight to the goal. This is not the case for the Coastal algorithm which ignores the uncertainty and strives to remain in the proximity of corners and edges until sufficiently close. This approach could potentially lead to unnecessary travel cost which could otherwise have been avoided.

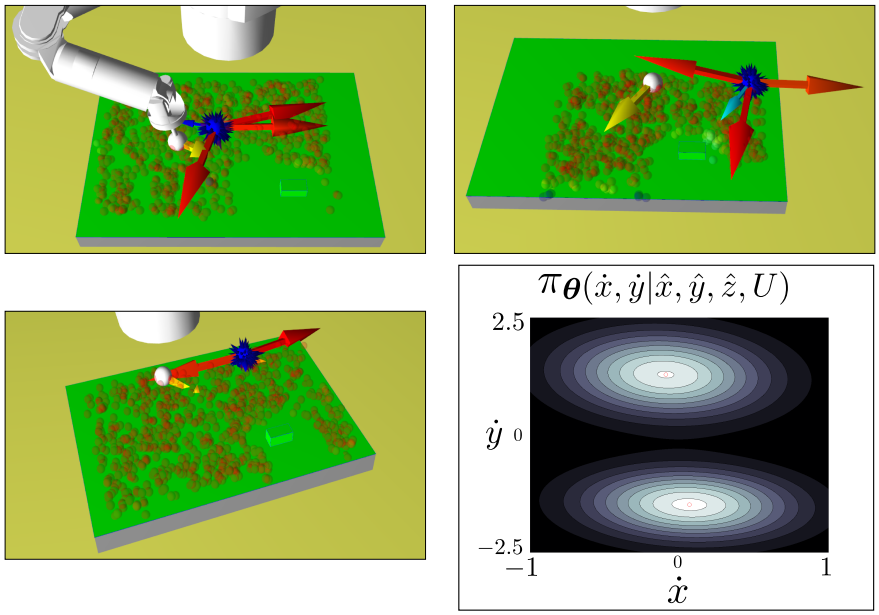


Figure 3.12: Illustration of three different types of modes present during the execution of the task where the robot is being controlled by the learned GMM model. The white ball represents the actual position of the robot's end-effector. The blue ball represents the believed position of the robot's end-effector and the robot is acting according to it. The blue ball arrows represent modes. Colours encode the model's weights given by the priors $w^{[k]}$ after conditioning (but not re-weighted as previously described). The spectrum ranges from red (high weight) to blue (low weight). *Top left:* Three modes are present, but two agree with each other. *Top right:* Three modes are again present indicating appropriate ways to reduce the uncertainty. *Lower left:* Two modes are in opposing directions. No flipping behaviour between modes occurs since preference is given to the modes pointing in the same direction as the robot's current trajectory. *Lower right:* GMM modes when conditioned on the state represented in the lower left figure. The two modes represent the possible directions (un-normalised).

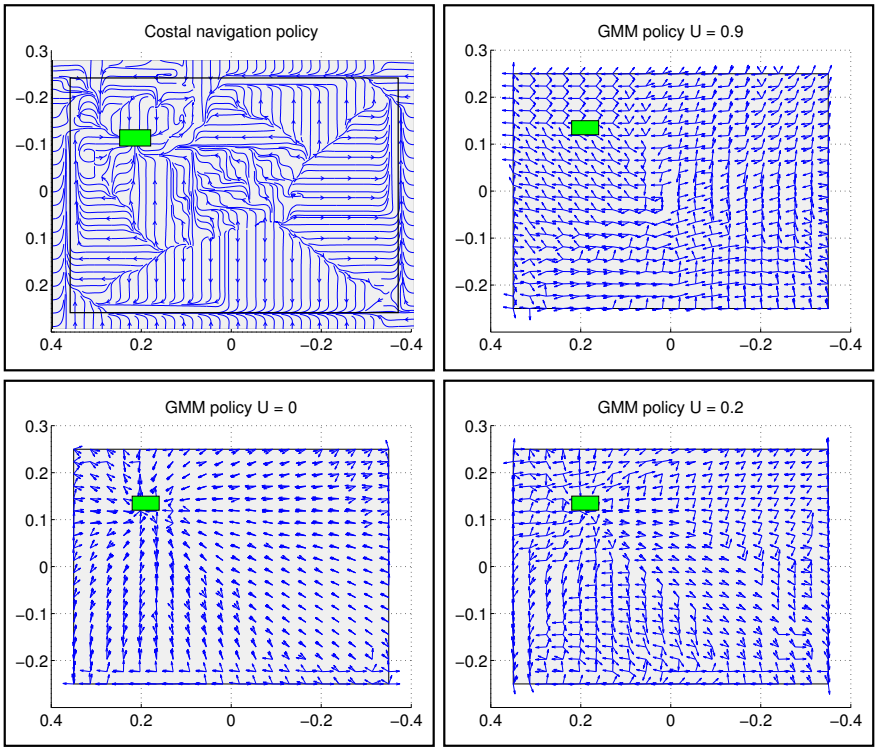


Figure 3.13: Illustration of the vector field for the Coastal and GMM policy. *Top Left* Coastal policy, there is only one possible direction for every state at any time, the values of λ_2 in the cost function were set experimentally. *Others:* The GMM policy for three different levels of uncertainty. For each point multiple actions are possible which is reflected by the number of arrows (only the first three most likely actions). As the uncertainty decreases the policy becomes less multi-modal, but remains around the edges and corners. Note that once certain of being close to an edge there is a possibility to go either straight to the goal or stay close to the edge and corners.

3.6.3 Distance efficiency & Uncertainty

We seek to distinguish the most efficient method in terms of two metrics, the distance (in meters) taken to reach the goal and the level of uncertainty upon arriving at the goal. We report results on 5 different search experiments in which we compare the Greedy, GMM and Coastal Navigation algorithms. The Hybrid was not fully considered since it is a heuristic combination of the Greedy and GMM methods.

In the first experiment, the true and believed locations of the end-effector were drawn uniformly from the original start distribution (Figure 3.3, *top right*) reflecting the default setting. The initializations (both real and believed end-effector locations) for the remaining 4 experiments were chosen in order to reflect particular situations which highlight the differences and drawbacks between each respective search method. For the first experiment (Uniform search experiment), a 100 trials were carried out in which the end-effector position and belief were initialized uniformly. As for the other 4 search experiments, 40 separate runs were carried for each of the three algorithms.

Table 3.2 reports the mean and variance of the distance taken (in meters) to reach the goal for each search method for all 5 experiments. We report on an Analysis of Variance (ANOVA) to test that all experiments were significantly different from one another as were the searches. We test the null hypothesis, H_o, that there is no statistical difference between the 5 search experiments. Before performing the ANOVA, we verified that our dependent variable, distance [m] taken to reach the goal, follows a normal distribution for all methods and all experiments (a total of $5 \times 3 = 15$ tests), an assumption which is required by an ANOVA analysis. A Kolmogorov-Smirnov test was performed on each experiment and associated search method. A total of 11/15 searches rejected the null hypothesis with a significance level of less than 5% (p-value < 0.05).

In Table 3.3 we report the p-values and F-statistics for an ANOVA on the 5 different experiments where our null hypothesis is that all experiments produce statistically the same type of search. For all experiment types the p-value is extremely small, below a significance value of 1% (p-value < 0.01) which indicates that we can safely reject the null hypothesis and accept that all experiments

Experiment	**Greedy**	**GMM**	**Coastal**
Uniform	1.54 (0.46)	0.99 (0.14)	1.13 (0.57)
#1	3.02 (0.36)	1.82 (0.23)	3.44 (1.50)
#2	0.80 (0.01)	1.41 (0.14)	0.94 (0.01)
#3	1.14 (0.08)	1.80 (0.17)	2.14 (0.81)
#4	0.75 (0.04)	1.34 (0.07)	0.68 (0.01)

Table 3.2: Mean distance and (variance) taken to reach the goal for 3 methods in 5 experiments. The grey shaded entries correspond to the results of the search algorithm which obtained the fastest time to reach the goal in each type of experiment/search.

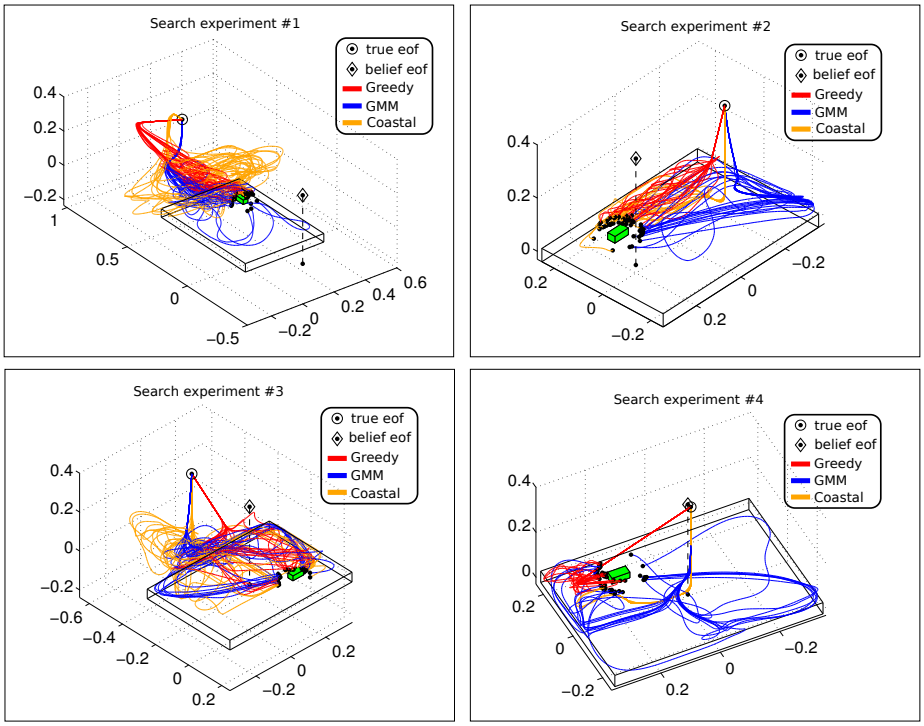


Figure 3.14: Four search initializations, from *top left* to *bottom right* we refer to them as #1-4. The circle indicates the true starting point of the end-effector (eof), whilst the triangle is the initial believed location of the eof. The initialisation in #1 was chosen such that the true and believed eof locations were at opposite sides of the table. This setting was selected to highlight the draw back in methods which do not take into account uncertainty. The second initialisation #2, reflects the situation where once again there is a large distance between true and believed location of the eof. However this time both are above the table. The starting points in #3 are a variant on #1 with the difference being that the believed eof position is above the table whilst the true eof location is not. The last experiment #4 was a setup which would be favourable to algorithms that are inclined to be greedy. Both true and believed eof locations are close to one another.

search method	Uniform	#1	#2	#3	#4
p-value (F)	2e-06 (14)	5e-07 (19)	7e-11 (36)	4-06 (15)	4e-16 (67)

Table 3.3: ANOVA tests the null hypothesis that all search experiments produced the same type of search with respect to the distance taken to reach the goal. All the p-values are extremely small which indicate that the null hypothesis can safely be rejected.

p-value (F)	Greedy vs GMM	Greedy vs Coastal	GMM vs Coastal
Uniform	3.59e-08 (30)	3.32e-04 (13)	1.90e-01 (2)
#1	5.80e-08 (46)	1.88e-01 (2)	4.58e-06 (28)
#2	3.60e-08 (47)	4.68e-04 (14)	4.54e-06 (28)
#3	3.57e-07 (37)	2.07e-05 (23)	1.25e-01 (2)
#4	6.70e-10 (64)	1.58e-01 (2)	6.34e-13 (107)

Table 3.4: ANOVA between paired search methods. The first column gives an indication of the probability that both the Greedy and GMM searches are statistically the same (the null hypothesis). This was rejected with a tolerance of below %1. In the second column, Greedy vs Coastal searches #1 and #4 are statistically closer than the rest with a p-value threshold of 10% required to be able to reject the null hypothesis. In the third column the uniform and #3 are not statistically different and would require a higher threshold on the p-value to be so.

produced very different searches, which is important for a comparative study.

As the first ANOVA only indicated that the experiments produced different searches, we also performed a second ANOVA test between the paired search methods to confirm that the methods themselves are statistically different. Table 3.4 illustrates the difference between the individual search methods for each experiment. It was found that most search algorithms produced significantly different searches (p-value < 0.01) with the exception of the GMM and Coastal algorithm for the Uniform and #3 experiment (p-value < 0.1). However the GMM and Coastal trajectories for the #3 experiment appear to be quite different when the trajectories are off the table's surface, see Figure 3.14 *(Bottom left)*, but share similar characteristics such as edge following behaviour.

From our ANOVA analysis we conclude that the behaviour exhibited by the three search strategies is significantly different. This is certainly the case for the Greedy and GMM methods, even though in certain situations the Greedy and Coastal policies display similar behaviour such as in experiment #1. The reason for this is that both the Greedy and Coastal policies start in a situation where there are no salient features available and their polices take the true end-effector location to an even more feature deprived region. In this situation the GMM policy is the clear winner with respect to the distance taken to reach the goal.

In experiment #2, both Greedy and Coastal policies perform equally well, and usually perform faster than the GMM model if the true and believed locations of the end-effector remain on the surface of the table. Otherwise if this is not the case, they will both reduce the uncertainty in a very inefficient way as the modes will often change during the search. This leads to the believed position (most likely state, $\hat{x}_t$) varying greatly, resulting in an increased time before the uncertainty has been narrowed down sufficiently for a contact to occur with

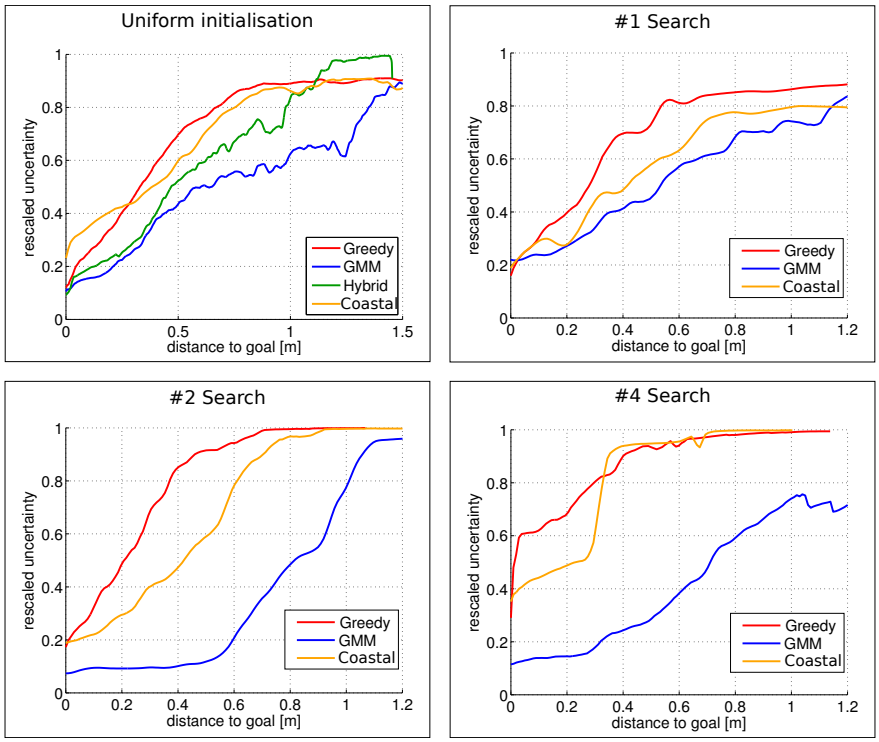


Figure 3.15: Reduction of the uncertainty for the Uniform, #1, #2 and #4 experiment, the expected value is reported *Top left*: Uniform initialisation, expected uncertainty for the Greedy (red), GMM (blue), Hybrid (green) & Coastal (orange) search strategies. *Top right:* Experiment #1. *Bottom left:* Experiment #2. *Bottom right:* Experiment #4.

the table (or simply by chance).

Figure 3.15 shows the normalised uncertainty with respect to the distance remaining to the goal for all experiments, (#3 is excluded being similar to the #2).

The results show which methods actively minimise the uncertainty and which methods find the goal whilst being more dependent on chance. For all the reported experiments the GMM (learned from human searches) reaches a lower expected uncertainty than all other search algorithms. For the Uniform and #1 search experiment, all methods reach the same final uncertainty level. However, for the #2 and #4 experiments, the GMM reaches the goal with significantly lower uncertainty. It is inferred that the GMM model actively minimises the uncertainty which is also reflected in the distance it takes to reach the goal in comparison with the other methods.

While the Greedy (#2) and Coastal (#4) are faster than the GMM method, Table 3.2, both have a far higher level of uncertainty at the arrival which leads to the assumption that chance has a non-negligible effect on their success.

3.7 Conclusions

In this work we have shown a novel approach in teaching a robot to act in a partially observable environment. Through having human volunteers demonstrate the task of finding an object on a table, we recorded both the inferred believed position of their hand and associated action (normalised velocity). A generative model mapping the believed end-effector position to actions was learned, encapsulating this relationship. As speculated and observed, multiple strategies are present given a specific belief. This can be interpreted as the fact that humans act differently given the same situation.

The behaviour recorded from the human demonstrations, encoded as set of expected sensations, showed the presence of trajectories which both remained near to the edge and corner features but also trajectories which remained at a distance. Risk-prone and risk-averse behaviour was further confirmed by the overlap of the risk factor of Human and GMM generated trajectories with that of the Greedy risk factor. According to the feature-based factor, more than 70% of the human search trajectories were considered to be risk-averse whilst 93% according to the Risk factor. Similarly the GMM search trajectories showed to be 89-88% risk-averse.

In terms of the comparative study, the GMM controller is more adapted to dealing with situations of high uncertainty and accounts for it better than Greedy or Coastal planning approaches. This is evident in the experiment where the believed position and true position of the end-effector were significantly far apart and distant from salient areas. Future questions of scientific value to be addressed are to which extent do humans follow the reasoning of a Markov Decision Process in a partially observable situation where the state space is continuous (the problem has been partially addressed in Baker et al. (2011) for discrete states and actions).

A drawback of the PbD-POMDP approach is that the quality of the learned policy is dependent on the abilities of the human teacher. If the teacher is consistent and goal directed (on average) then the transferred policy will be adequate, if however the human is suboptimal at performing the task, then the resulting policy will be poor. An autonomous way of evaluating the quality of the demonstrations whilst learning a policy is necessary. In the next chapter, "Chapter 4", we demonstrate that by introducing a cost function and using a Reinforcement Learning approach we can account for poor demonstrations and increase the quality of the policy.

Chapter 4

Peg in hole

In Chapter 3, we demonstrated that we could learn a search policy and successfully transfer it to a robot apprentice from demonstrations of human teachers for a task consisting of locating a wooden object on a table. In search tasks of this nature, our approach is based on the observation that intuition and knowledge exhibited by the human teachers contains a good balance between exploration and exploitation actions which can then be encapsulated in a generative Gaussian Mixture Model (GMM) and be subsequently used as a control policy. This chapter is to appear in de Chambrier and Billard (2016a).

The approach is satisfactory if the objective is to reproduce the same distribution of the behaviours exhibited by the teachers. However for learning an optimal or approximate optimal policy with a unique behaviour the PbD-POMDP approach will not necessarily result in an efficient policy. For the GMM models both the good and the bad search strategies exhibited by the human teachers. If the task is difficult and many possible solutions exist, such as in the previously discussed blindfolded search task in Chapter 3, many demonstrations will be required for search patterns to emerge and be encoded in the GMM. Otherwise the GMM needs to be combined with another policy as previously shown (Hybrid GMM-Greedy policy).

GMM does not discriminate between behaviours, as the Expectation-Maximisation (EM) algorithm used to learn the GMM policy ignores the quality of the demonstrated data. The EM algorithm does not contain a cost or reward function, encoding the objective of the task, which is common practice in Planning and Reinforcement Learning (RL). In the case of a difficult task where there are mostly bad demonstrations, the GMM search policy will reproduce suboptimal behaviour. Additionally there may be no single teacher who demonstrates satisfactory behaviour (in terms of achieving the task), however by combining different demonstrated components from individual teachers a good search strategy can be extracted.

To overcome the above mentioned limitations, we introduce a binary cost function as a means of ranking the human teachers' demonstrations. For instance a reward of 100 is given when the task is achieved and 0 otherwise. To this end, we combine our PbD-POMDP approach with an Actor-Critic Reinforcement Learning (RL) framework which is close to Fitted-Value Iteration

(FVI) and other Experience replay methods. We refer to this new method as RL-PbD-POMDP. Our objective is to avoid the noisy explorative rollouts, a weakness common to all RL approaches, and only rely on the data provided by the human teachers. Autonomous random exploration has two limiting factors for the application of RL to robotic systems.

Firstly it is time consuming and is typically only applicable where an exhaustive exploration of the entire state or parameter space is feasible, such as in the inverted pendulum or mountain cart type problems. The universal exploration method, used throughout RL, is state independent (sometimes state dependent) white noise which results in an entire exploration of the state space. This is neither practical nor feasible for the type of search problems we are considering. Secondly in this search problem as we are using a physical robotic system the exploration cannot be random as this would be dangerous.

We analyse our RL-PbD-POMDP approach on a power-socket Peg-in-Hole (PiH) search task and compare it to PbD-POMDP. In this task, human teachers must demonstrate to a robot apprentice how to search for and connect a plug to a power socket, see Figure 4.1 (*Left*). The first part of the task, the search for the socket, is similar to the table wooden-block setup in the previous chapter. However the connection of the plug to the power socket, the PiH component, requires a higher level of precision. In Figure 4.1 (*Right*) the robot reproduces the behaviour demonstrated by the teacher.

Figure 4.1: Peg-in-hole (PiH) search task. *Left:* A teacher is wearing ear defenders (to impede any hearing) and a blindfold. He first searches for the socket's location and then attempts to establish a connection. Force and torque information is obtained from an ATI 6-axis force torque sensor at the end-effector of the tool held by the teacher. *Right:* The KUKA LWR4 robot equipped with the same force torque sensor and plug reproducing the teacher's demonstrated behaviour.

4.1 Outline

- 4.2 Background

  We review aspects of the literature related to the Peg-in-hole problem and Actor-Critic Reinforcement Learning with an emphasis on Fitted methods (also known as Batch or Experience replay), which we adapt to our GMM policy.

- 4.3 Experiment methods
  We detail the Peg-in-Hole search task, the number of participants (teachers) which provide demonstrations, the type of data which is recorded and the representation of the human's location belief.

- 4.4 Learning Actor and Critic
  We detail the representation of the Actor and the Critic and how they are learned in a Fitted Policy Evaluation framework.

- 4.5 Control architecture
  The learned behaviour is transferred to a 7 Degree of Freedom (DoF) articulated robot, named KUKA LWR4. We detail the hybrid position-force controller and dynamical field modulation heuristic which is used in combination with the learned behaviour policy from the human teachers.

- 4.6 Results
  We perform a set of 5 simulated experiments to test the generalisation, the importance of data provided by the human teachers and the performance against simple search approaches to find the location of the socket. We further perform 3 experiments on the real robotic platform to test the generalisation of the learned policy on different power sockets.

- 4.7 Conclusion
  The RL-PbD-POMDP policy achieves an important improvement over the previous PbD-POMDP approach by learning a value function over the belief space using approximate dynamic programming (part of FVI) and using it to update the parameters of our GMM policy. We evaluate the ability of the policies to generalise to novel sockets and different socket locations, both in simulation and on the KUKA LWR robot. The RL-PbD-POMDP approach consistently proves to be better. More importantly the RL-PbD-POMDP approach performs significantly better when it is used on the worst teacher's demonstrations, which mitigates the **original assumption** that teachers have to be consistently efficient at the task.

4.2 Background

4.2.1 Peg-in-hole

The Peg-in-Hole (PiH) task is one of the most widespread step in industrial assembly and manipulations processes, with examples including the assembly of vehicular transmission components Siddharth and Branicky (2001) and valves Cheng and Chen (2014). To be successful, the estimated position of the robot's end-effector and workpiece must be precise. Typically, the clearance between

peg and plug is very small leaving little room for error. As a result, variations in the assembly's components in combination with position uncertainty can result in either jamming during the insertion process or in failure (hole not found). This created a need for adaptive search and insertion policies for PiH, which has been driving research in this area.

From the literature, we identified the different components in PiH solutions. All approaches use to some extent a vision system to estimate the position of the workpiece. For instance in Meeussen et al. (2010) a PR2 is equipped with a checkerboard to facilitate pose estimation of the plug with respect to a power outlet whose position is extracted through a vision processing pipeline. An initial connection is attempted by "visual servoing" which is successful 10% of the time. Given an estimate of the workpiece's position, a common approach is to follow either a blind increasing spiral Cartesian trajectory or parametrised policies which guarantee that all positions on the workpiece have been visited. In Meeussen et al. (2010), if the PR2 initially fails to connect the plug to the socket a spiralling outward motion is carried out with 2mm increments which obtains an overall success rate of 95%. For this approach to be applicable to a generic robot, it would require the addition of an external camera and checkerboard to the robot in question which might be cumbersome. In our work we consider a vision free system.

Another approach (which has been confined to academic circles) follows the data driven Programming by Demonstration (PbD) framework. Teleoperated or kinesthetic demonstrations by a human teacher are recorded and a policy is learned and fine-tuned so as to reproduce the same (F)orce/(T)orque profile as that demonstrated by the human teacher.

In Yang et al. (2014) the authors learn a PiH policy for the Cranfield benchmark object. A vision system obtains the pose parameters of the object whilst a human teacher demonstrates trajectories, through teleoperation, in the frame of reference of the object. A time-dependent policy represented with Dynamic Movement Primitives (DMP) (Schaal et al., 2004) encodes the recorded Cartesian end-effector pose. In Nemec et al. (2013), a F/T profile is encoded separately by a regressor parameterised by radial basis functions. Successive refinements of the DMP policy are achieved through using force feedback to adapt the parameters of an admittance controller. This results in the policy having similar force profiles to the human teachers. Further applications based on this method have been performed by Abu-Dakka et al. (2014) with the incorporation of a disturbance rejection policy. Reproducing exactly the same force torque profile for the full trajectory which is encoded in a time dependent dynamical system might be unnecessary as the force torque profile is predominantly useful during the final stage of the PiH task, where the insertion can cause jamming. The force torque information can be used to rectify this problem (Kronander, 2015, Chap. 5). A hybrid control paradigm (Fisher and Mujtaba, 1992) can also be used to control the sensed force feedback with the environment. We make use of the

hybrid control paradigm in this work in combination with a time-independent dynamical system.

Reinforcement learning has also been used in combination with DMP to learn PiH policies. In Kalakrishnan et al. (2011) an DMP policy is initialised with kinesthetic demonstrations of opening a door and picking up a pen. The recorded Cartesian trajectories are encoded in a parameterised DMP policy and augmented with a F/T regressor profile. A reward function is designed, encoding desirable properties of the F/T profile such as smoothness and continuity. After 110 trials the policy was found to be 100% successful. In Gullapalli et al. (1994) a 18 dimensional input (sensed position, previous position and force) and a 6 dimensional output (linear and angular velocity) neural network is learned by associative reinforcement learning. During the learning process the plug is randomly positioned within the vicinity of the hole. After a 100 executions and updates, the policy was shown to be successful and was able to generalise across different geometries and clearances. Our work is similar in its approach, however we will not be considering autonomous rollouts common in RL, but will rely solely on the initial data provided by human teachers.

All the above policies were learned from human demonstrations and encoded by a regressor function and optimised to reproduce a desired F/T profile. Other approaches to the PiH problem are predominantly based on heuristic search mechanisms and compliant controllers.

In Siddharth and Branicky (2001) different blind search policies are analysed for the insertion of a spline toothed hub into a forward clutch. The state space is discretised into points so that the distance between two neighbours is smaller than the clearance of the hole, which is known as a spray point coverage. Different search strategies are evaluated which ensure that all the points are visited. It is found that paths following concentric circles gradually spiralling inwards are the most effective method for finding the hole. This concentric circle search strategy has been applied in many PiH tasks. For instance in Bdiwi et al. (2015), a PiH heuristic policy was developed to connect a 5-pin waterproof industrial charger to an electric socket. The authors estimated the pose of the socket through a vision system and used a force controller in combination with a blind spiral search policy to achieve a connection and demonstrated their approach to be reliable. These blind search strategies do not consider actual state uncertainty and only work well when the plug or peg is within the vicinity of the socket. In our work we consider no visual information which leads to high state uncertainty making the direct application of such blind search methods ill-suited.

In Park et al. (2013) the authors observe that humans lack the precision and sensing accuracy of robotic systems, but nevertheless, are more proficient than robots at PiH. The authors state that when humans try to connect a square plug to a socket, they rub the plug against the socket's outlet without looking. It is thought that the inherent compliance in humans' motor control

is the key to our success at PiH tasks kook Yun (2008). The authors introduce an Intuitive Assembly Strategy (IAS) inspired by the above observation which does not require the hole to be precisely localised. The IAS search strategy is based on compliant spiral motion and the execution of the search trajectory is performed with a hybrid force/position controller. We also have observed that humans are good at accomplishing such tasks and we exploit this in our own PiH policy. We further consider different types of geometric objects whilst only considering haptic information.

The spiral strategy is widely used in industrial applications due to its simplicity, however, it is a blind search method. Another approach when dealing with the assembly process consists of fine-tuning parameters of predefined policies. In Cheng and Chen (2014) the authors develop an online Gaussian Process policy optimisation of an assembly task. They demonstrate that by learning the dynamical model of the task during execution, it is faster than offline methods, such as Design of Experiment (DOE) or Genetic Algorithms.

4.2.2 Actor-Critic & Fitted Reinforcement Learning

In our PiH-search task, to learn a POMDP policy, we consider RL approaches which naturally handle continuous belief-state and action spaces, such as policy search/gradient methods. Chapter 2 gives an overview of such policy search methods. However, policy gradient methods are time consuming when the number of parameters is large compared with the state space, Zhao et al. (2015). This results from the large variance of the policies' gradient, making the stochastic gradient ascent learning slow. In Chapter 3, the learned PbD-POMDP policy has over 80 Gaussian functions, each of dimension 7, and we expect the number of parameters of this RL policy to be of the same order. So instead we opt for an Actor-critic (AC) approach which has been reported and proven to be more efficient in terms of the number of episodes necessary to achieve convergence (Grondman et al., 2012), as the variance of the gradient of the policy's parameters have a smaller variance compared with actor methods (policy gradient). This results in a faster learning of the policy.

Actor-critic (Sutton and Barto, 1998, Chap. 6.6) is an RL approach in which the actor (policy) and critic (value function) have separate parameterization and can be represented by different functions, for instance the value function could be a decision tree and the policy a neural network. The advantage of an AC is that the policy can be chosen such that it is computationally efficient in evaluating actions and the value function does not necessarily have to have the same input space as the policy.

The policy gradient theorem (Sutton et al., 2000) states that if the regressor functions of both the actor and critic share the same basis functions (also called *compatible* features) and their parameters are linear, then an unbiased estimate

of the policies' gradient can be obtained. The drawback of this approach is that both the value function and policy have to be defined over the state-action space. This is restrictive and in addition it has been shown that Function Approximators (FA), such as linear models or neural networks, when combined with temporal difference learning, can diverge (Boyan and Moore, 1995).

As we are working in belief-space we seek a framework in which both the actor and critic can have their own parameterisation whilst guaranteeing convergence of the value function. All value function approximators, such as tile coding, state aggregation, k-nearest-neighbour, locally weighted averaging and grid discretisation are all averagers and are guaranteed to converge in model based RL (Gordon, 1995) when used with temporal difference learning. The key idea presented in Gordon (1995) and which lead to the increase in popularity of batch and fitted methods, is to separate the Bellman and value function in a synchronous update, that is, to first compute the Bellman update for all the sample states and then fit a value function via standard supervised learning techniques. The extension to a model-free approach with a kernel function approximator (locally weighted averaging, the kernel is a Gaussian function) known as Kernel-Based Approximate Dynamic Programming (KBDP) (Ormoneit and Glynn, 2002) has proven to be globally optimal in a continuous-space framework. This led to the wider application of Batch RL methods such as Fitted Value Iteration (FVI) (Bou-Ammar et al., 2010) and Fitted Q-Iteration (FQI) (Ernst et al., 2005a; Neumann and Peters, 2009b) (Q-approximator is a random forest ensemble). By remembering all the state transition pairs and by applying multiple synchronous Dynamic Programming (DP) and function approximation updates, the problem of diverging value function approximators is resolved.

Retaining all the data makes it in practice easy to apply function approximators which are not averagers, such as neural networks, to RL problems. A successful example was Neural Fitted Q-Iteration (NFQI) (Riedmiller, 2005) which uses a multi-layer perceptron to represent the Q-function for the cart-pole and mountain car problems and shows rapid convergence to optimal policies. It has since been used in many extensions, Peters and Schaal (2008a), Agostini and Celaya (2010). This has resulted in the application of more sophisticated regression methods such as the increasingly popular Deep Learning methods which are known as Deep Fitted Q-iteration (DFQ) (Lange and Riedmiller, 2010) (used to learn visual control policies) and with recent work including learning to play ATRI and ping-pong games Mnih (2015), Hausknecht and Stone (2015) (reviewed in Chapter 2).

The reader is referred to Busoniu et al. (2011) for a literature review which includes a taxonomy of Batch RL methods and to (Wiering and van Otterio, 2012, Chap 2) for a concise description Batch RL beginning at its origins, how it became popular with Fitted RL approaches and its continuation into Deep Learning.

The RL-PbD-POMDP framework which we will use in this chapter is also

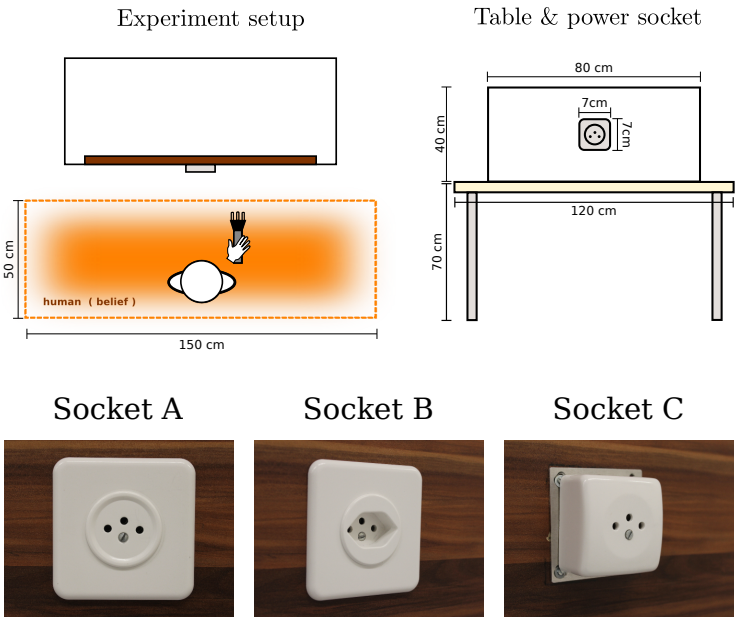


Figure 4.2: The experimental setup. *Top-left:* A participant (human teacher) is blindfolded and placed within the orange rectangular area always facing the wall. *Top-right:* Dimensions of the the wall and socket. *Bottom:* Three different power sockets, only socket A and B are used for data collection, socket C is purely used for evaluating the generalisation of the learned policy.

based on a Fitted approach, however we avoid performing the expensive maximisation over the continuous actions space, as in the FVI and FQI approaches, by fitted policy evaluation followed by policy improvement. We use a Gaussian Mixture Model to parameterise the policy and a Locally Weighted Regression (LWR) as the value function approximator.

4.3 Experiment methods

Figure 4.2 (*Top-left*), illustrates the PiH-search experiment setup. The orange area represents the teachers starting area and is assumed prior knowledge. The sockets are always positioned at the center of a fake wall (wooden plank) which is clamped to a table, see Figure 4.2 (*Top-right*) for an illustration.

We consider one type of plug, Type J[1], and three different power sockets. Power *socket A*, has a ring around its holes, *socket B* has a funnel, which we hypothesize should make it easier to connect, and *socket C* has a flat elevated surface. See Figure 4.2 (*Bottom*) for an illustration.

The human teacher holds the plug which is attached to a cylindrical handle with an ATI 6 axis force torque sensor (Nano25[2]) to provide **raw** wrench $\phi \in \mathbb{R}^6$ measurements. We define the **actual** measurement to be a function of the raw wrench, $\tilde{y}_t = h(\phi_t)$, which is a binary feature vector. The feature vector

[1] http://www.iec.ch/worldplugs/typeJ.htm
[2] http://www.ati-ia.com/products/ft/sensors.aspx

Figure 4.3: Human holding the cylinder plug holder, which is equipped with OptiTrack markers.

encodes whether a contact is present and the direction in which it occurs, which is discretized to the four cardinalities.

On top of the cylinder there is a set of markers used by a motion capture system OptiTrack[3] (which has millimeter tracking accuracy), see Figure 4.3, to measure both linear, $\dot{x} \in \mathbb{R}^3$, and angular velocity, $\omega \in \mathbb{R}^3$, at each time step which is recorded at a rate of 100 Hz. The force and torque information from the ATI sensor is recorded at the same rate.

In this task, the human's location belief is represented by a probability distribution function. The participants' (teachers) initial belief is assumed to be uniformly distributed as depicted in orange area of Figure 4.2 and that all subsequent beliefs can be inferred from the measured velocity and measurements provided by the ATI and OptiTrack sensors. The following section describes how the belief can be represented, computed and compressed.

Belief state

For the task at hand, the belief probability density function, $p(x_t|y_{0:t}, \dot{x}_{1:t})$, is a Point Mass Filter (PMF) (Bergman and Bergman, 1999, p.87), which is a Bayesian filter. It is parametrised by a set of grid cells containing valid probabilities and is recursively updated by the application of a **motion** $p(x_t|x_{t-1}, \dot{x}_t)$ and **measurement** $p(y_t|x_t)$ model. The motion model updates the position of the probability density function and subsequently increases the uncertainty of the position. The measurement model indicates areas of the state space from which a measurement $\tilde{y}_t$ could have originated. In Figure 4.4 (*Bottom-right*) we illustrate the likelihood when an edge is sensed.

A PMF is chosen to represent the believed location of the plug as the sensing likelihoods are non-Gaussian and lead to multi-modal distributions. A PMF is able to capture such non-Gaussianity whilst remaining fully deterministic (which is not the case for a particle filter).

The probability density function $p(x_t|y_{0:t}, \dot{x}_{1:t})$ is high dimensional and thus it is impractical to directly learn a statistical policy $\pi_\theta : p(x_t|y_{0:t}, \dot{x}_{1:t}) \rightarrow \dot{x}_t$ without some form of compression. One possibility would be E-PCA (Roy and Gordon, 2003) which extracts a set of representative basis features. Although

[3] http://www.optitrack.com/

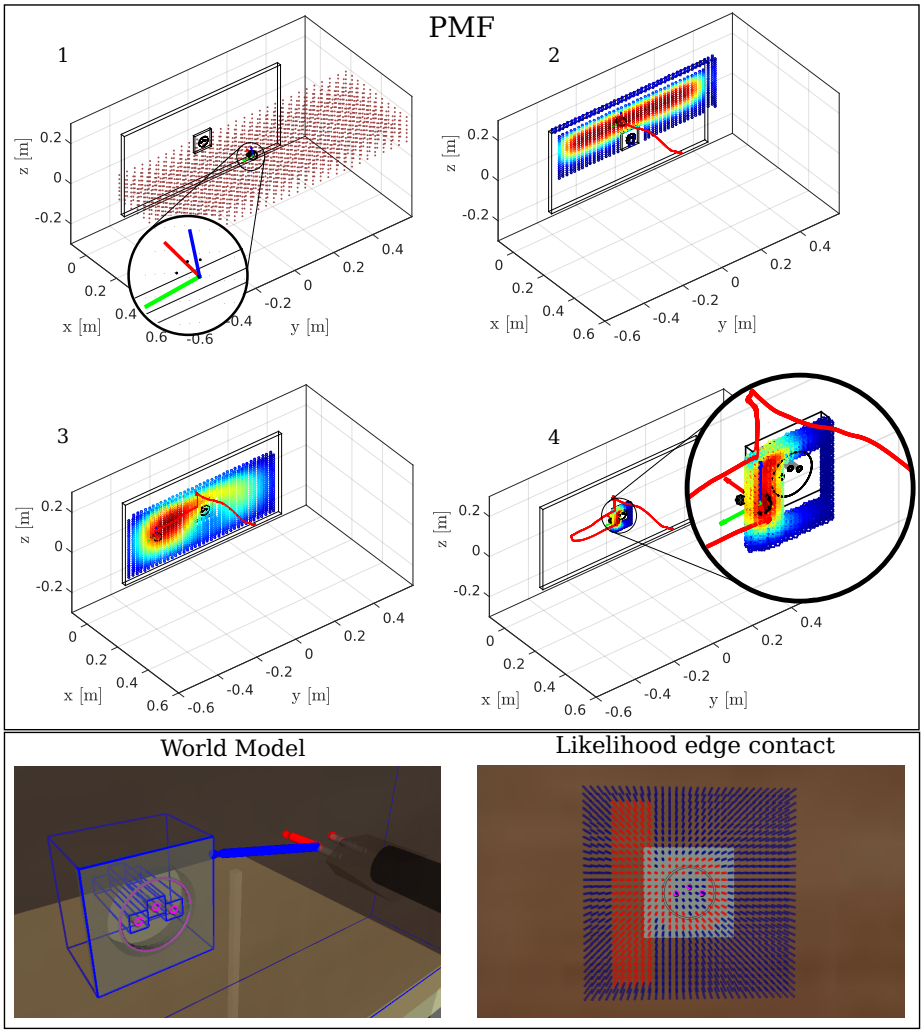


Figure 4.4: *Top box:* Point Mass Filter (PMF) update of a particular human demonstration. (1) Initial uniform distribution spread over the starting region. Each grid cell represents a hypothetical position of the plug. The orientation is assumed to be known. (2) First contact, the distribution is spread across the surface of the wall. The red trace is the trajectory history. (3) motion noise increases the uncertainty. (4) The plug is in contact with a socket edge. *Bottom box*: **World model**: The plug is modelled by its three plug tips and the wall and sockets are fitted with bounding boxes. **Likelihood**: The plug enters in contact with the left edge of the socket. As a result, thmotioe value of the likelihood in all the regions, x_t, close the left edge take a value of one (red points) whilst the others have a value zero (blue points) and areas around the socket's central ring have a value of one.

elegant this method requires a discretisation of the belief space which is computationally expensive. Instead we choose to compress the pdf to a belief space vector composed of the maximum a posteriori (most likely state), as in Chapter 3 page 57.

Each participant's demonstration results in a dataset $\mathcal{D} = \{\dot{x}^{[i]}_{1:T}, \omega^{[i]}_{1:T}, \phi^{[i]}_{1:T}, b^{[i]}_{1:T}\}$, where the upper index $[i]$ references the ith search trajectory (also one execution of the task or one episode) and subscript $1:T$ denotes the time steps during the trajectory from initialisation $t = 1$ until the end $t = T$.

4.3.1 Participants and experiment protocol

To perform the PiH search tasks we recruited 10 student volunteers to be teachers (all male Master's and PhD students). The participants were aged between 24 and 30 with an average age of 26 years and a standard deviation of 2.4 years. Each participant carried out 30 demonstrations of the PiH search-task and each session lasted approximately 50 minutes and never exceeded one hour. The 10 participants were divided equally in two groups, A and B. Each member of group A began by performing 15 PiH searches with socket A, followed by a 10 minute break, finishing with an additional 15 searches with socket B. The members of group B performed the same protocol starting with socket B and ending with socket A. Figure 4.5 summarises a walk through of the experiment. The only exclusion criteria was the inability of the subject to accomplish the task. All participants gave written consent for taking part in this study.

The next section describes in detail the protocol for the search task:

1. Participant signs a form of consent before starting the experiment.

2. Each participant is given the opportunity to familiarise themselves with the environment and become comfortable in wearing the sensor deprivation apparatus. During this time the participant is allowed to practice connecting the plug to the socket whilst standing within its vicinity.

3. Once the participant feels sufficiently ready to carry out the task to the best of his ability, the experimenter proceeds to disorient him through the usage of swivel chair. The disorientation process takes 30 seconds and includes both translation and rotation motions. After disorientation, the participant is signalled to stand up. The participant is reminded that he is facing the direction of the wall and that his starting location is within the orange rectangular area demarcated on the floor. He is then signalled by a light touch to the shoulder that he can start the task.

4. At task completion, the subject is once again disoriented and the process is repeated a total of 15 times. After 15 trials, the subject is given a 10

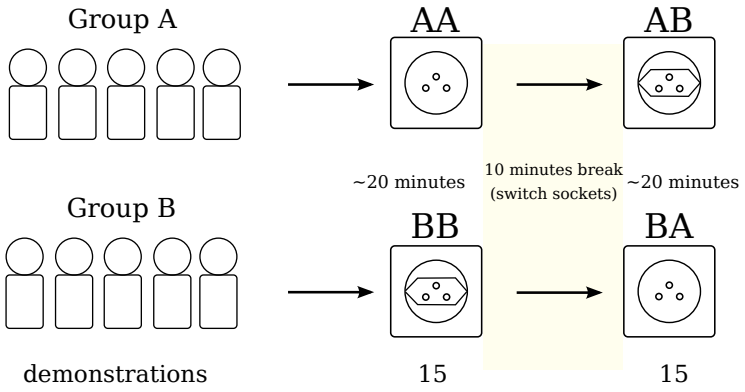


Figure 4.5: Experiment protocol. The participants are divided in two groups of 5, Group A begins with socket A and after a short break repeats the task with socket B. The same logic holds for Group B. For each socket 15 executions of the task are recorded.

minute break whilst the experimenter changes the type of socket (A or B). A participant of group A will now continue with socket B. Similarly a participant of group B will continue after the break with socket A.

Each participant carried out a total of 30 PiH-search experiments, giving a total of 300 demonstrations.

Preliminary results

Both groups A and B took 9 ± 10s to find the socket's edge, regardless of the socket type. This is to be expected since the sockets are at the same location. It took a further 8 ± 7s on average for group B to connect socket B and 12 ± 10s on average for group A to connect socket A. As we can see this is not a straight forward task when considering the sensory deprivation. See Figure 4.6 (*Bottom*) for the time taken to connect the plug to the socket. In Appendix A.1 we report the results of a non-parametric statistical analysis on the time taken to connect the sockets and we find that it takes 4 seconds more to connect socket A than socket B. This is somewhat expected as socket B has a funnel which can help to contain the subject to within the vicinity of the holes.

As connecting socket A is more difficult we will be **using only these demonstrations** as training data to learn a policy. Both socket B and C will be used solely to evaluate the generalisation of the policy.

4.4 Learning Actor and Critic

In our approach we learn two data driven policies. The first policy maps from belief space to linear velocity $\pi_{\boldsymbol{\theta}_1} : b \mapsto \dot{x}$ and the second from wrench to angular velocity, $\pi_{\boldsymbol{\theta}_2} : \phi \mapsto \omega$. We chose to learn the belief policy $\pi_{\boldsymbol{\theta}_1}$ in a Actor-Critic RL framework and the wrench policy $\pi_{\boldsymbol{\theta}_2}$ directly from the demonstrated

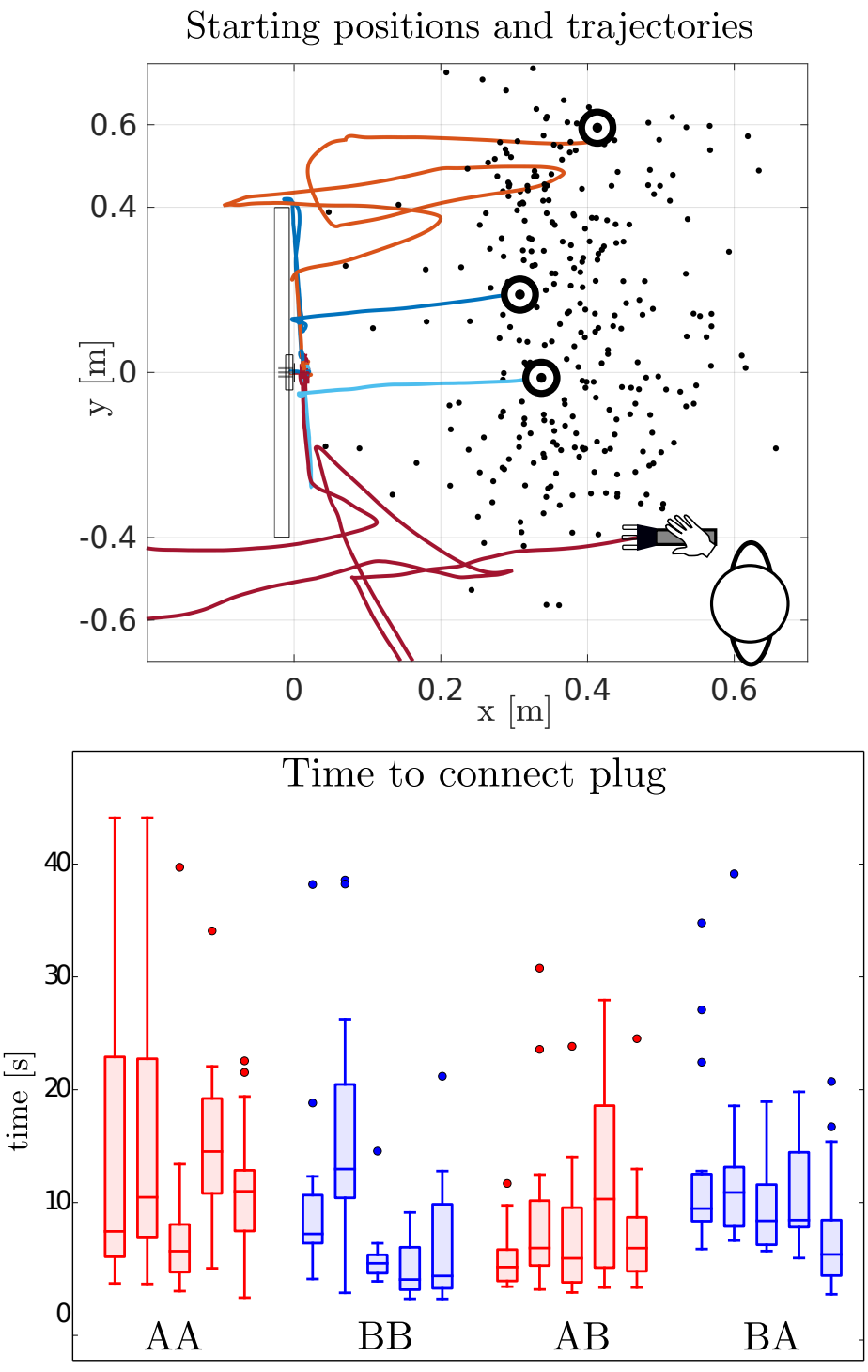


Figure 4.6: *Top*: Black points represent the starting position of the end-effector for all the demonstrations. Four trajectories are illustrated. *Bottom:* Time taken for the teachers to accomplish the PiH once the socket is localised. Group A and B are depicted in red and blue. The second letter indicates which socket is used, see Figure 4.5.

data as was done in (Kronander, 2015, Chap. 5), which proved to be efficient in overcoming jamming during the PiH. A POMDP solver's objective is to find a policy (Actor), $\pi_{\boldsymbol{\theta}_1} : b \mapsto \dot{x}$, which maximises the value function (Critic) $V^{\pi_{\boldsymbol{\theta}_1}} : b \mapsto \mathbb{R}$ for an initial belief, b_0. The value function is the expected reward over an infinite time horizon.

$$V^{\pi_{\boldsymbol{\theta}_1}}(b_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}_1}} \left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \right\} \in \mathbb{R} \tag{4.4.1}$$

In an Actor-Critic setting, the Temporal Difference (TD) error, $\delta_t^\pi = r_{t+1} + \gamma V^\pi(b_{t+1}) - V^\pi(b_t)$, of the value function (the Critic) is used as a learning signal to update simultaneously itself and the actor (the policy) (Sutton and Barto, 1998, Chap. 6). In our method, two separate policies are learned, one for the linear velocity and the other for the angular velocity. The orientation is kept constant until the start of the connection of the plug to the socket. The angular velocity policy used and learned only during the insertion of the plug to the socket where it is necessary to control the orientation to avoid jamming.

4.4.1 Actor & Critic

A generative model of the angular velocity and wrench $\pi_{\boldsymbol{\theta}_2}(\omega, \phi)$ and a generative model of the linear velocity and belief state $\pi_{\boldsymbol{\theta}_1}(\dot{x}, b)$ are learned. The structure of $\pi_{\boldsymbol{\theta}_1}$ is the same as in Chapter 3 Section 3.5 on page 61. In both cases we use the Bayesian Information Criterion to determine the number of Gaussian functions. In the next section, we will show how the parameters of $\pi_{\boldsymbol{\theta}_1}$ can be adapted by the value function of the Critic.

The Critic (Equation 4.4.1) evaluates the performance of the current policy. It is the expected future reward given the current belief state and policy. In our method a reward of $r = 0$ is received at each time step until the goal (plug-socket connection) is achieved, where a reward of 100 is given, $r_T = 100$. Given the continuous nature and dimensionality of the belief space we use Locally Weighted Regression (LWR) (Atkeson et al., 1997) as a function approximator of the value function, $V^\pi(b)$. LWR is a memory-based non-parametric function approximator. It keeps a set of input-target pairs $b \mapsto r$ as parameters. When a value b is queried, a set of p neighbouring points are chosen from the input space and are weighted according to a distance metric. The predicted output is given by a weighted least square of the p points. Equation 4.4.2 is the distance function used where D is a diagonal matrix.

$$W_{i,i} = \exp\left(-\frac{1}{2} (b - b_i)^{\mathrm{T}} D^{-1} (b - b_i) \right) \tag{4.4.2}$$

A new value is queried according to Equation 4.4.3,

$$V^\pi(b) = b\,(B^{\mathrm{T}} W B)^{-1} B^{\mathrm{T}} W \mathbf{r} \tag{4.4.3}$$

where $B = (b_1, \ldots, b_p)^{\mathrm{T}} \in \mathbb{R}^{(D \times p)}$, $W \in \mathbb{R}^{(p \times p)}$ is a diagonal matrix, $\mathbf{r} = (r_1, \cdots, r_p)^{\mathrm{T}} \in \mathbb{R}^{(p \times 1)}$

4.4.2 Fitted Policy Iteration

Policy evaluation

To learn the value function we make use of batch reinforcement learning (Ernst et al., 2005a), also known as Experience replay. This is an offline method which applies multiple sweeps of the Bellman backup operator over a dataset of tuples $\{(b_t^{[i]}, \dot{x}_t^{[i]}, r_t^{[i]}, b_{t+1}^{[i]})\}_{i=1,\cdots,M}$ until the Bellman residual, $||V_{k+1}^{\pi}(b) - V_k^{\pi}(b)||$, converges.

Algorithm 1: Fitted Policy Evaluation

input : ϵ, $\{(b_t^{[i]}, r^{[i]}, b_{t+1}^{[i]})\}_{i=1,\cdots,M}$
output: $\hat{V}_k^{\pi}(b_t)$

while $||\hat{V}_{k+1}^{\pi}(b) - \hat{V}_k^{\pi}(b)|| > \epsilon$ **do**
  $\hat{V}_{k+1}^{\pi}(b_t) = \text{Regress}(b,\ r_t + \gamma \hat{V}_k^{\pi}(b_{t+1}))$

Batch RL methods are used by a broad spectrum of research to learn policies. Most of them have focused on learning the Q-value function directly (Fitted Q-Iteration) (Neumann and Peters, 2009a; Ernst et al., 2005a; Riedmiller, 2005). Although this solves the control problem it requires discretisation of the action space or assumes quantifiable actions, as the Q-Bellman backups, $\hat{Q}(b_t, \dot{x}_t) \leftarrow \gamma \max_{\dot{x}_{t+1}} \hat{Q}(\dot{x}_{t+1}, b_{t+1})$, require an optimisation over the action space, $\dot{x}_{t+1}$, to find the best applicable action. Given the dimensionality and continuity of our problem we opt for an on-policy evaluation method which requires multiple *policy evaluation* and *policy improvement* iterations to achieve an optimal policy. In order for the RL-PbD-POMDP and PbD-POMDP to be comparable, we will only be performing one iteration of policy evaluation and improvement, hence Algorithm 1 is applied only once to the dataset.

Policy improvement

In our offline approach the value function of the belief state, $\hat{V}^{\pi}(b)$, is estimated until convergence and then used to update the actor. This offline batch method has the advantage that no divergence can occur during the learning process.

We update the Actor policy given the Critic value function through a modification of the Maximisation step in Expectation-Maximisation (EM) for Gaussian Mixture Models. We refer to this modification as Q-EM which is strongly

related to a Monte-Carlo EM-based policy search approach (Deisenroth et al., 2011, p.50).

The reward of a demonstrated trajectory (one episode) is given by the discounted return, Equation 4.4.4,

$$R(b^{[i]}, \dot{x}^{[i]}) = \sum_{t=0}^{\mathrm{T}^{[i]}} \gamma^t \, r(b_t^{[i]}, \dot{x}_t^{[i]}) \tag{4.4.4}$$

where the index i stands for the ith episode. All policy gradient approaches seek to find a set of actor parameters $\boldsymbol{\theta}$ which will maximise the expected reward, equivalent to maximising Equation 4.4.5,

$$\begin{aligned} J(\boldsymbol{\theta}) &= \mathbb{E}_{p_{\boldsymbol{\theta}}}\{R\} \\ &= \sum_{i=1}^{N} \underbrace{\left(\prod_{t=0}^{T^{[i]}} \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \right)}_{p_{\boldsymbol{\theta}}(\tau_i)} R(\tau_i) \end{aligned} \tag{4.4.5}$$

where $\tau_i = \{(\dot{x}_0, b_0), \cdots, (\dot{x}_T^{[i]}, b_T^{[i]})\}$ are the state-action samples of the ith episode. To find the parameters which maximise the cost function, $\arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, its derivative is set to zero. As this cannot be done directly, we maximise the logarithmic lower bound of the cost function which results in Equation 4.4.6, see Appendix A.2 for the derivation.

$$\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \, Q^{\pi_{\boldsymbol{\theta}'}}(\dot{x}_t^{[i]}, b_t^{[i]}) \tag{4.4.6}$$

Setting the derivative of Equation 4.4.6 to zero and solving for the parameters $\boldsymbol{\theta} = \{w, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ leads to a Maximisation update step of EM which is weighted by $Q^{\pi_{\boldsymbol{\theta}'}}$. The parameters $\boldsymbol{\theta}'$ are those used to generate a set of rollouts whilst $\boldsymbol{\theta}$ are the new parameters being estimated. We use the Critic's TD error as a substitute for Q^{π}. Assuming that our estimated value function, $\hat{V}^{\pi}$, is close to the true value function V^{π}, the TD error δ^{π} is an unbiased estimate of the advantage function, Equation 4.4.7 (see Appendix A.4).

$$A^{\pi}(\dot{x}_t, b_t) = Q^{\pi}(\dot{x}_t, b_t) - V^{\pi}(b_t) = \delta_t^{\pi} \tag{4.4.7}$$

Using the advantage function as means of policy search is popular with methods such as Natural Actor Critic (NAC) Peters and Schaal (2008b).

Each state-action sample has an associated weight, $\delta_t^{\pi} \in \mathbb{R}$, where $\delta_t^{\pi} > 0$ means that the state action-pair lead to an increase in the value function and $\delta_t^{\pi} < 0$ lead to a decrease in the value function. The data log-likelihood is re-weighted accordingly, giving more importance to data points which lead to a gain. Since the Q-EM update steps cannot allow negative weights, the TD error is rescaled to be between 0 and 1.

The reader is referred to Appendix A.3 for the Maximisation update step of Q-EM for a GMM parameterization of the policy.

2D example fitted policy evaluation and improvement

To illustrate the mechanism of fitted policy evaluation and improvement, we give a 2D example of its application, see Figure 4.7. The *Top-left* subfigure depicts 10 trajectories demonstrated by two teachers going from start (white circle) to goal (orange star) state. The optimal path is a straight line passing in between two obstacles. Neither teacher demonstrated the optimal straight path.

In the *Bottom-left*, a GMM is fitted $\pi_{\boldsymbol{\theta}}(\dot{x}, x)$ to the teachers' data, using the standard EM-algorithm. Taking the policy to be the output of Gaussian Mixture Regression (GMR) $\mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\}$ we obtain different behaviours than those demonstrated by the human teachers. The GMR averages the different modes encoded by the Gaussian functions which results in a mixing of the original demonstrated behaviours. No trajectories of the GMR policy truly replicate the demonstrated behaviour.

In the *Top-right* subfigure, we apply fitted policy evaluation to the original demonstrated data (discount factor $\gamma = 0.99$ and reward $r = 1$ when the goal is reached and zero otherwise) and compute the value function.

The *Bottom-right* subfigure illustrates the GMM policy learned with the Q-EM algorithm. As the advantage function $A^{\pi}(x, \dot{x})$ is highest along the start-goal axis, data points following this gradient will have a higher weight. This results in a policy with better rollouts (closer to the optimal path) than the trajectories generated by the policy learned via standard EM.

Belief state fitted policy evaluation

Returning to the PiH-search task with socket A, the Fitted Policy Evaluation (FPE) Algorithm 1 is applied to the demonstrations. In Figure 4.8 we illustrate the value function of the most likely state after the FPE algorithm converges. As expected, the value function is high closest to the socket and around the axis $z = 0$ and $y = 0$. When policy improvement via Q-EM is applied the Gaussian functions of the GMM will favour these locations.

In Figure 4.9 we illustrate the best and worst trajectories in terms of the accumulated value function. We can see that the best trajectories (red) tend to be aligned with the socket (star position in front of socket), whilst the worst trajectories are towards the edges of the wall and tend to follow spiralling movements.

We learned two policies, one solely from the original human demonstrations which we call GMM and the second which is the result of one iteration of fitted policy evaluation and improvement which we call Q-EM. We first detail the robot control architecture in the next section before we compare the GMM and

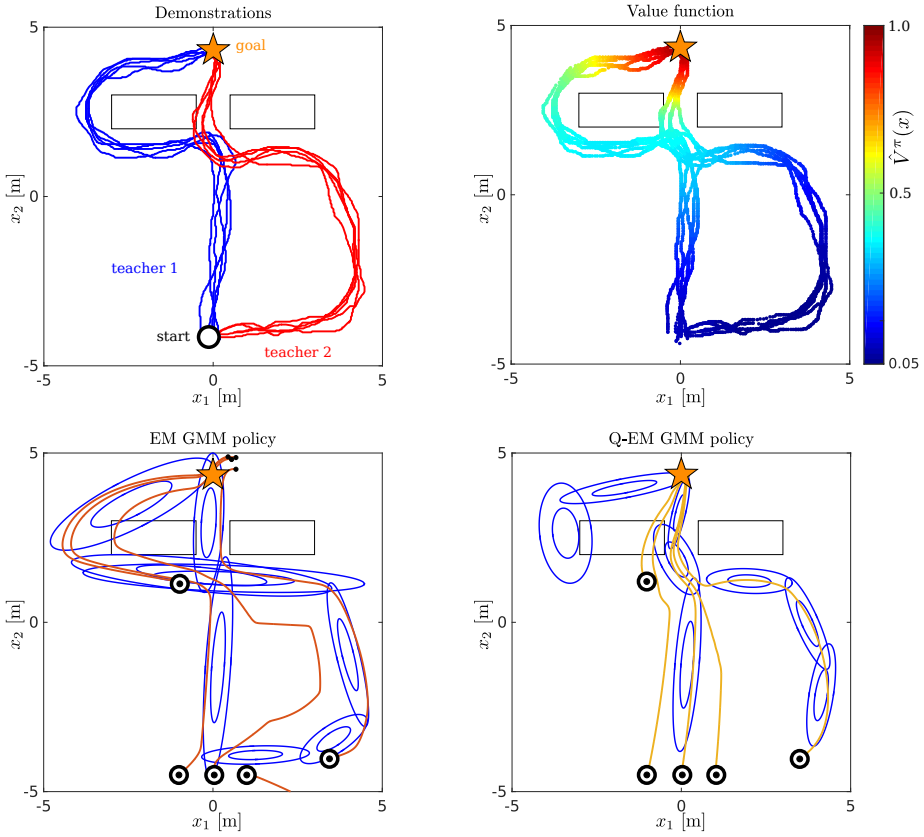


Figure 4.7: Fitted policy evaluation and improvement example. *Top-left:* The goal of the task is to reach the goal state. The first teacher (blue) demonstrates five trajectories which contours the obstacle in front of the goal. The second teacher (red) demonstrates 5 trajectories which initially deviate from the goal before passing between the two obstacles. *Bottom-left:* The EM algorithm is used to fit a GMM to the teachers' original data. The marginal $\pi_{\boldsymbol{\theta}}(x)$ is plotted in blue and trajectories generated by the policy $\mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|x)\}$ in orange. *Top-right Policy Evaluation:*. Value function after fitted policy evaluation terminated, the reward function is binary, $r = 1$ at the goal and zero otherwise, and a discount factor $\gamma = 0.99$ is used. *Bottom-right Policy Improvement:* the GMM is learned with the Q-EM algorithm in which each data point's weight is proportional to the advantage function.

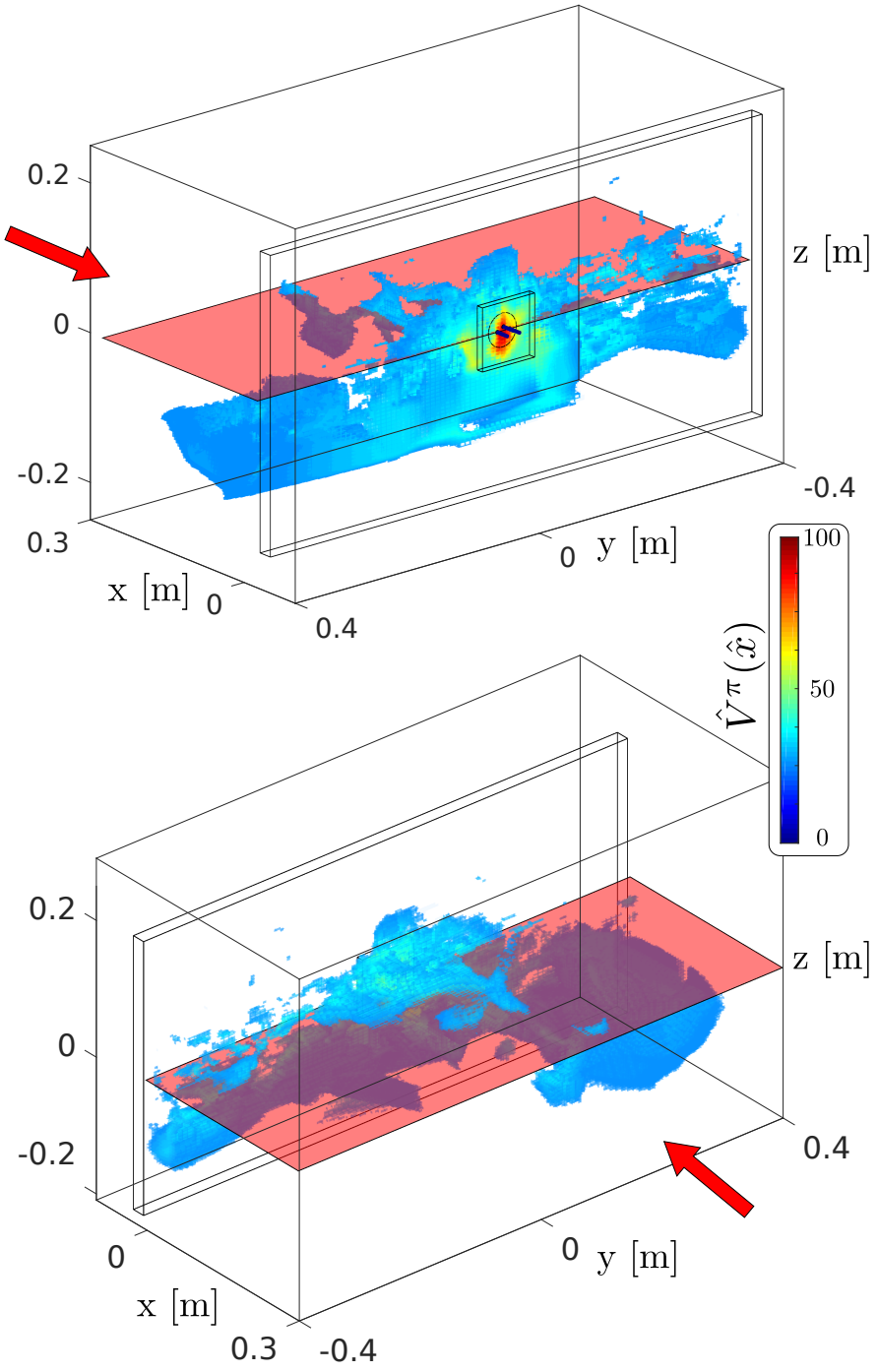


Figure 4.8: LWR value function approximate $\hat{V}^{\pi}(\hat{x})$ for the most likely state $\hat{x}$. The red plane is to help visualise where the value function is above and below the axis $z = 0$. Only states with values above 0.25 are plotted. The red arrow indicates the heading of the human teacher when performing the search task. The discount factor was $\gamma = 0.99$ and the variance of the kernel variance of 1 [cm], which was set experimentally.

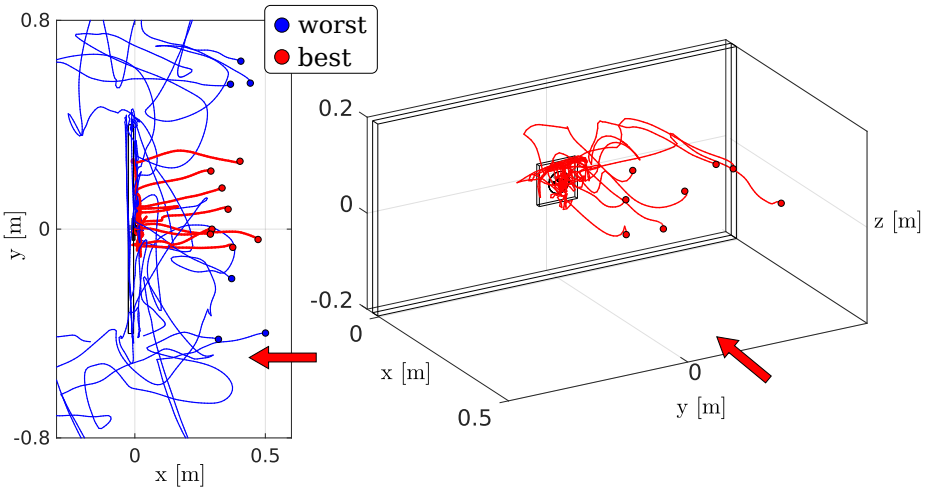


Figure 4.9: Best and worst trajectories. The red demonstrated trajectories are the best in terms of the amount of value function gain whilst the blue are the worst. The red arrow indicates the teacher's heading. The blue trajectories tend towards the sides of the wall as the initial starting position is on the boarders of the wall. The red trajectories are centred along the y-axis of socket and tend to move in a straight line towards the wall whilst aligning themselves with the axis $z = 0$.

Q-EM policies in the section 4.6.

4.5 Control architecture

As detailed in section 4.4.2, a Gaussian Mixture Model was learned for both linear and angular velocity, although only the linear control policy is active until the plug is within the socket's hole, as the orientation is constant. The direction to search is given by the conditional, Equation 4.5.1,

$$\pi_{\boldsymbol{\theta}}(\dot{x}|b) = \sum_{k=1}^{K} w_{\dot{x}|b}^{[k]}\, g(\dot{x}; \boldsymbol{\mu}_{\dot{x}|b}^{[k]}, \boldsymbol{\Sigma}_{\dot{x}|b}^{[k]}) \tag{4.5.1}$$

which is a distribution over the possible normalised velocities. The function $g(\cdot)$ is a multivariate Gaussian function parameterised by mean $\boldsymbol{\mu}_{\dot{x}|b}^{[k]} \in \mathbb{R}^{(3\times 1)}$ and Covariance $\boldsymbol{\Sigma}_{\dot{x}|b}^{[k]} \in \mathbb{R}^{(3\times 3)}$. The subscript $\dot{x}|b$ indicates that the parameters are the result of the conditional. The reader is referred to Calinon et al. (2010), Sung (2004) for a detailed derivation of the conditional of a GMM. The learned model is multi-modal, as different search velocities are possible in the same belief state. Figure 4.10 illustrates the multi-modal vector fields of the conditional, Equation 4.5.1. In autonomous dynamical systems control, the velocity is obtained from the expectation of the conditional, Equation 4.5.1. However, the expectation which is a weighted linear combination of the modes, could result in unobserved behaviour or no movement if the velocities cancel out. As a result we use a modified version of the expectation operator which favours the current direction,

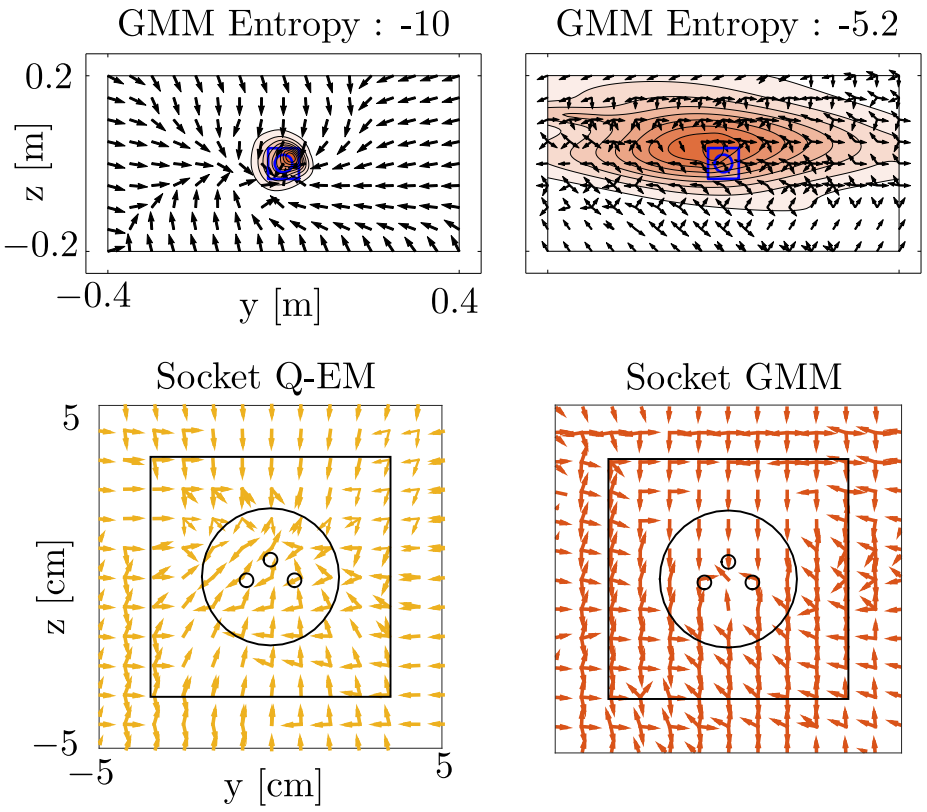


Figure 4.10: Q-EM and GMM policy vector fields. *Top*: The GMM policy is conditioned on an entropy of -10 and -5.2. For the lowest entropy level, most of the probability mass is close to the socket area since this level corresponds to very little uncertainty; we are already localised. We can see that the policy converges to the socket area regardless of the location of the believed state. For an entropy of -5.2 we can see that the likelihood of the policy is present across wall. The vector field directs the end-effector to go towards the left or right edge of the wall. *Bottom*: The entropy is marginalised out, the yellow vector field is of the Q-EM and orange of the GMM. The Q-EM vector field tends to be closer to a sink and there is less variation.

the same steps in Chapter 3 Section 3.5.3 on page 64 are taken to generate a control signal for the robot.

4.5.1 Robot Implementation

The GMM policy outputs a linear velocity which is normalised, $\underline{\dot{x}} \in \mathbb{R}^{(3\times 1)}$. The amplitude of the velocity is computed separately and modulated according to sensed forces on the end-effector. This search task is haptic and the end-effector of the robot is always in contact with the environment. To make the robot compliant with the environment we use an impedance controller in combination with a hybrid position-force controller. A hybrid controller targets a sensed force F_x, in the x-axis, of 3 Newtons (N) . The y and z velocity components of the direction vector are given by Equation 3.5.7 on page 64. This is insufficient for the robot to reliably surmount the edges of the socket, hence the

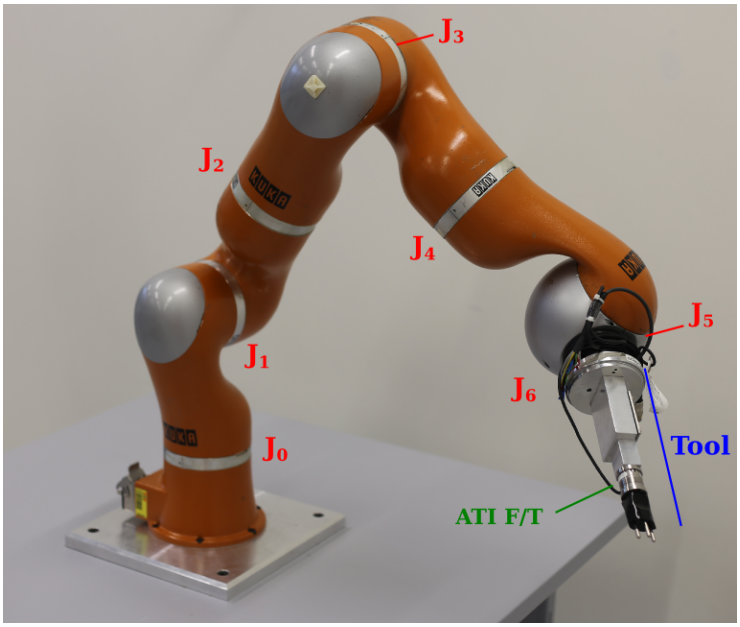


Figure 4.11: The KUKA LWR is a 7 Degree Of Freedom (DoF) robot, we illustrate in red each joint, which is controlled at a rate of 1kHz via an ethernet cable. The KUKA API provides a command interface to the stiffness, damping, position and torque variables of each joint. The tool is a peg holder equipped with an ATI force/torque sensor.

vector field of the GMM is modulated in y and z-axis, Equation 4.5.2.

$$\underline{\dot{x}} = R_y(c(F_z) \cdot \pi/2) \cdot R_z(c(F_y) \cdot \pi/2) \cdot \underline{\dot{x}} \tag{4.5.2}$$

where R_y and R_z are (3×3) rotation matrices around the y and z-axis, and $c(F) \in [-1, 1]$ is a truncated scaling function of the sensed force. When a force F_z of 5N is sensed, a rotation of $R_y(\pi/2)$ is applied to the original direction resulting in the robot getting over the edge. The direction velocity is always normalised up to this point. The amplitude of the velocity is a computed as in Chapter 3 Equation 3.5.8 on page 65.

The above procedure can control the general behaviour of the search but is insufficient for a successful implementation on a robotic system such as the 7 Degree of Freedom $q \in \mathbb{R}^7$ KUKA LWR, which we illustrate in Figure 4.11. The belief policy's outputs a linear velocity and the angular velocity is computed from a reference orientation which is constant. When the plug is to be connected to the socket, the angular velocity is the output of samples drawn from the conditional $\omega \sim \pi_{\boldsymbol{\theta}_2}(\omega|\phi)$. The steps to convert Cartesian velocities to torques to control the KUKA LWR are also the same as in Chapter 3. Figure 4.12 illustrates the complete control flow.

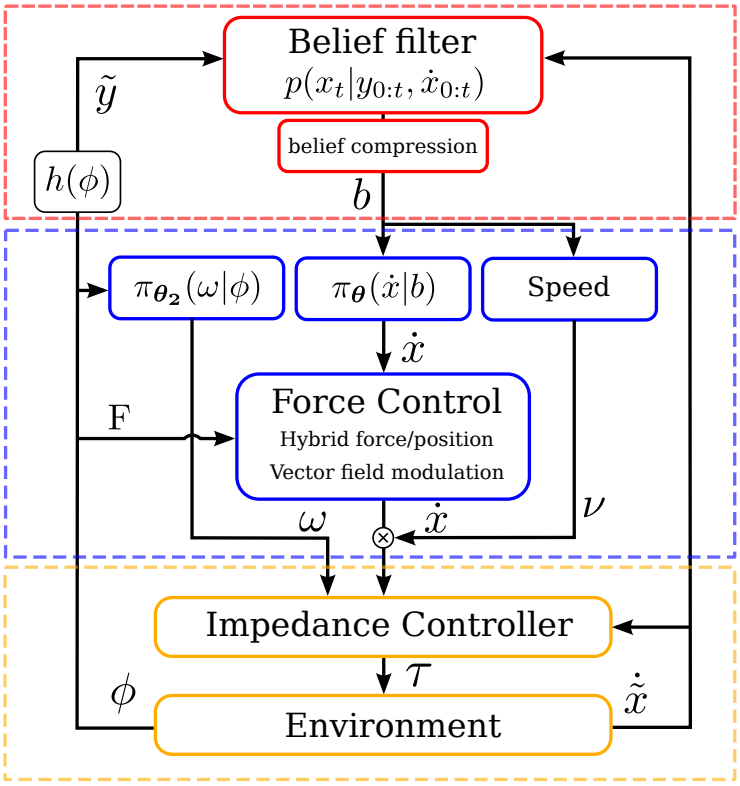


Figure 4.12: Control architecture. The PMF (belief) receives a measured velocity, $\dot{\tilde{x}}$, and a sensor measurement $\tilde{y}$ and is updated via Bayes rule. The belief is compressed and used by both the GMM policy and the proportional speed controller (see Chapter 3).

4.6 Results

We evaluate the following three aspects of the policy learned in our Actor-Critic framework:

1. **Distance taken to accomplish the goal** (connect plug to socket). We compare the Q-EM policy with a GMM policy learned through standard EM and a myopic Greedy policy. This highlights the difference between complicated and simplistic search algorithms and gives an appreciation of the problem's difficulty.

2. **Importance of data** provided by human teachers. We evaluate whether it is possible to learn an improved GMM policy from Greedy demonstrations. This policy which we call Q-Greedy is used to test whether indeed human demonstrations are necessary. We evaluate whether it is possible to obtain a good policy from the two worst teachers' demonstrations as not all teachers are necessarily proficient at the task in question and we want to test whether our methodology can be applied in these cases. We evaluate if we are able to obtain an improved policy from the worst two teachers.

3. **Generalisation**. We learn a policy to insert a plug into socket A which is located at the center of a wooden wall. We test the generalisation of the policy in finding a new socket location and whether the policy can generalise to sockets B and C, which were not used during the training phase.

We evaluate aspects 1) and 2) purely in simulation as finding the socket requires much less precision than establishing a connection and the physics of the interaction is simple. Aspect 3), the generalisation, is evaluated both in simulation, up to the point of localising the socket's edge, and on the KUKA LWR robotic platform for the connection phase of the task. The main reason for employing the robot is that the connection phase dynamics is complex and a simulation would be unrealistic. For the robot evaluation we consider the search starting already within the vicinity of the socket.

4.6.1 Distance taken to reach the socket's edge (Qualitative)

We consider three search experiments which we refer to as **Experiment 1, 2** and **3**, in order to evaluate the performance in terms of the distance travelled to reach the socket for the three search policies: GMM, Q-EM and Greedy. In these three experiments the task is considered accomplished when a search policy finds the socket's edge.

In **Experiment 1**, three starting locations are chosen: *Center*, *Left* and *Right*. See Figure 4.13, *Experiment 1*, for an illustration of the initial condition. This setup tests the effect of the starting positions. A total of 25 searches are carried out for each of the search policies.

In **Experiment 2**, two *Cases* are chosen in which the believed state (most likely state of the PMF) and the true position of the end-effector are relatively far apart. The location of the beliefs are chosen to be symmetric, see the Figure 4.13, *Experiment 2*. A total of 25 searches are carried for each of the two cases.

In **Experiment 3**, Figure 4.13, *Experiment 3*, the initial true starting positions of the end-effector are taken from a regular grid covering the whole start region, also used as the initial distribution for the human demonstrations. A total of a 150 searches are carried out for each of the three policies. This experiment compares the search policies with the human teachers' demonstrations.

We evaluate the performance of the three experiments in terms of the trajectories and their distribution in reaching the edge of the socket.

In **Experiment 1**, see Figure 4.13 *Experiment 1, second row* the results show a clear difference between the trajectories generated by the GMM and Q-EM policies. The orange GMM policy trajectories go straight towards the wall, whilst the yellow Q-EM policy trajectories drop in height making them closer to the socket. The same effect can be seen in Experiment 2 (*second row*). The Q-EM trajectories follow a downward trend towards the location of the socket. The gradient is less as the initial starting condition is lower than in Experiment 1.

In **Experiment 2**, see Figure 4.13, *Experiment 2, second row*, the trajectories of the Greedy policy depend on the chosen believed location (most likely state of the PMF). There is no variance in the Greedy's trajectories until it reaches the edge of the red square, where the branching occurs as the believed location is disqualified. This happens as no sensation has been registered at the point when the believed location reaches the wall. The true location is in fact situated further away from the wall than the believed location.

In **Experiment 3**, see Figure 4.13 *Experiment 3, second row*, Human and GMM show similar distributions of searched locations. They cover the upper region of the wall and top corners, to some extent. These distributions are not identical for two reasons. The first is that the learning of the GMM is a local optimisation which is dependent on initialisation and number of parameters. The second reason is that the synthesis of trajectories from the GMM is a stochastic process.

For the Q-EM policy, the distribution of the searched locations is centred around the origin of the z-axis. The uncertainty is predominantly located in the x and y-axis. The Q-EM policy takes this uncertainty into consideration by restraining the search to the y-axis regardless of the starting position. The uncertainty is reduced when it is in the vicinity of the socket. The Greedy's policy search distribution is multi-modal and centred around the z-axis where

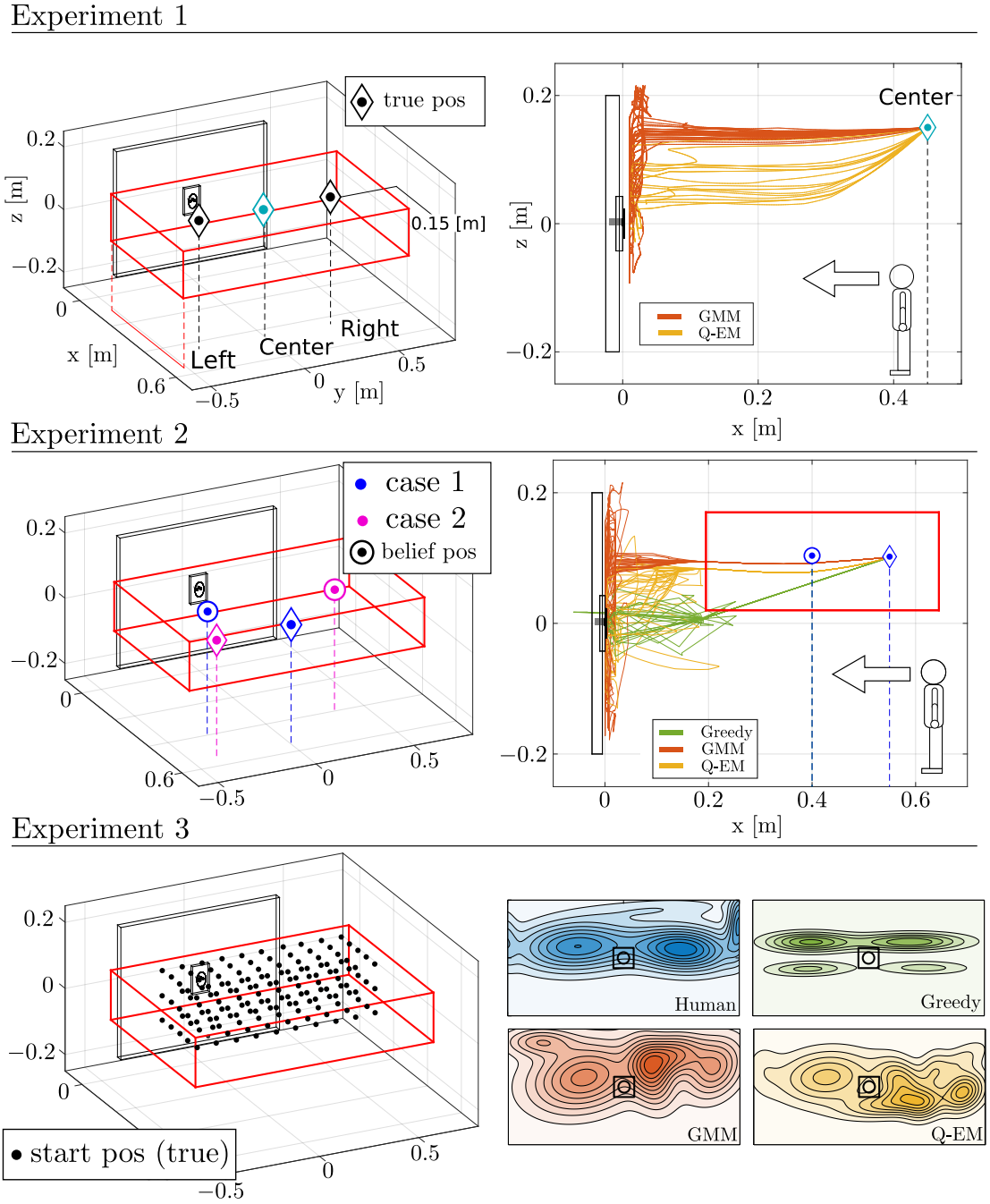


Figure 4.13: Three simulated search experiments. **Experiment 1:** Three start positions are considered: *Left*, *Center* and *Right* in which the triangles depict true position of the end-effector. The red cube illustrates the extent of the uncertainty. In the second row of Experiment 1, we illustrate the trajectories of both the GMM (orange) and Q-EM (yellow) policies. For each start condition a total of 25 searches were performed for each search policy. **Experiment 2:** Two cases are considered: *Case 1* blue, the initial belief state (circle) is fixed facing the left edge of the wall and the true location (diamond) is facing the socket. *Case 2* pink, the initial belief state (circle) is fixed to the right facing the edge of the wall and the true location is the left edge of the wall. In the second row, the trajectories are plotted for *Case 1*. **Experiment 3:** A 150 start locations are deterministically generated from a grid in the start area. In the second row, we plot the distribution of the areas visited by the true position during the search.

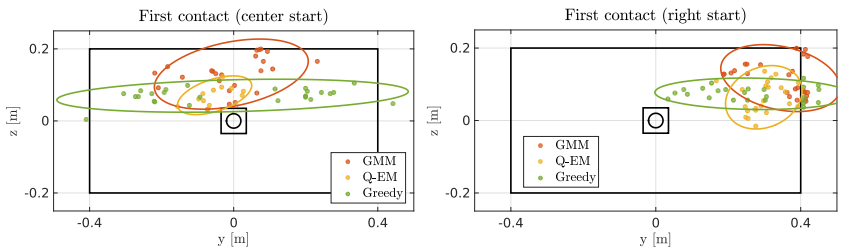


Figure 4.14: First contact with the wall, during experiment 1. (a) Contact distribution for initial condition "Center" . (b) Contact distribution for initial condition was "Right". The ellipses correspond to two standard deviations of a fitted Gaussian function.

the modes are above and below the socket. This shows that the Greedy policy acts according to the most likely state which changes from left to right of the socket, because of motion noise, resulting in left-right movements and little displacement. As a result the Greedy policy spends more time at these modes.

In Figure 4.14 (*Top-left*), we illustrate the distribution of the first contact with the wall during Experiment 1 for the *Center* initial conditions. The distribution of the first contact of the Greedy method is uniform across the entire y-axis of the wall. In contrast, the GMM policy remains centred with respect to the starting position and the Q-EM is even closer to the socket and there is much less variance in the location of the first contact.

4.6.2 Distance taken to reach the socket's edge (Quantitative)

In Figure 4.15 we illustrate the quantitative results of the distance taken to reach the socket for all three experiments. We performed a Kruskal-Wallis non-parametric test to assert if there is a significant difference between the means of the groups. In the figure we use the following notation: $(*)$: $p < 0.05$, $(**)$ $p < 0.01$, $(***)$ $p < 0.001$ and (ns) for not significant. In **Experiment 1**, for the *Center* initial condition, the Q-EM policy travels far less than the other search policies. Considering that the initial position of the search is 0.45 [m] away from the wall, the Q-EM policy finds the socket very quickly once contact has been established with the wall. For the *Right* and *Left* starting conditions both the GMM and Q-EM policies travel less distance to reach the socket, with a smaller variance when compared with the Greedy search policy. The Q-Greedy search policy was learned solely from demonstrations given by Greedy policy with state independent white noise as means of exploration.

In **Experiment 2**, Figure 4.15, the Q-EM search policy is the most efficient. For *Case 1* of Experiment 2, the initial most likely state is fixed to the left and the true position is facing the socket. As the belief is chosen to be to the left, upon contact with the wall the policy takes a left action since it is more likely to result in a localisation. On average this results in an exploration of the upper

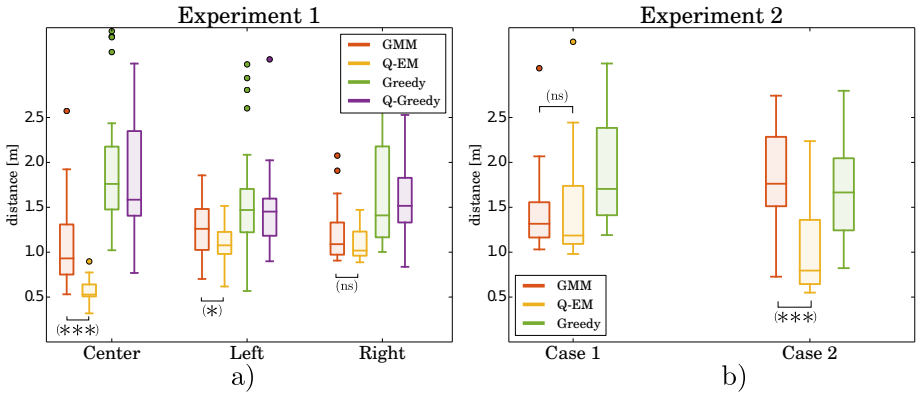


Figure 4.15: Distance travelled until the socket's edge is reached. a) Three groups correspond to the initial conditions: Center, Left and Right depicted in Figure 4.13, *top left*. The Q-EM method is always better than the other methods, in terms of distance. b) Results of the two initial conditions depicted in Figure 4.13, *top middle*, both the true position and most likely state are fixed. The Q-EM method always improves on the GMM. The Q-Greedy policy was learned from data provided by rollouts of a stochastic Greedy policy. Statistical significance is represented as follows: $(*)$: $p < 0.05$, $(**)$ $p < 0.01$, $(***)$ $p < 0.001$ and (ns) for not significant.

left area of the wall, which explains why *Case 1* of Experiment 2 performs worse than Experiment 1 for the *Center* initial condition. In *Case 2* however, where the true state is facing the left edge and the believed position is facing the right edge, less distance is taken to find the socket than for *Case 1*, Figure 4.15 (b). This improvement over *Case 1* is due to the true location of the end-effector being closer to an informative feature and results in a much faster localisation.

From **Experiment 3**, Figure 4.16, all four search policies travel less to find the socket's edge compared with the teachers' demonstrations. All search policies are better than the human teachers with the exception of group BA, which is performing the task with socket A. The Q-EM policy remains the best.

We have shown that under three different experimental settings the Q-EM algorithm is predominantly the best in terms of distance taken to localise the socket. The GMM policy learned solely from the data provided by the human teachers also performs well in comparison to the human teachers and Greedy policy. We made, however a critical assumption in order to be able to use our (RL-)PbD-POMDP approach. This **assumption** is that a human teacher is proficient in accomplishing the task. If a teacher is not able to accomplish the task in a repetitive and consistent way so that a search patterns can be encoded by the GMM, the learned policy will perform poorly. Next we evaluate the validity of this assumption and the importance of the training data provided by the human teachers.

4.6.3 Importance of data

We perform two tests to evaluate the importance of the teachers training data

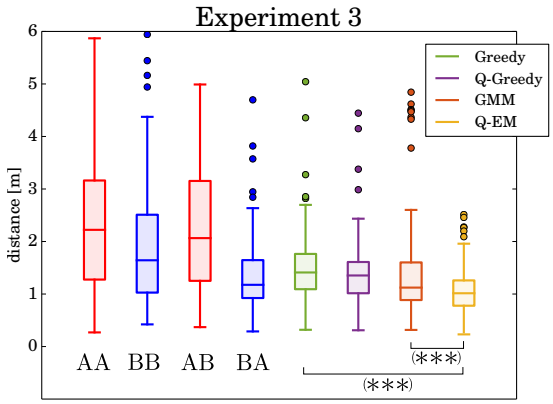


Figure 4.16: Distance travelled until the socket's edge is reached. Results corresponding to Experiment 3, Figure 4.13, *top right*. Again the Q-EM method is better, but at a less significant level.

for learning a search policy. Firstly we take the worst two teachers in terms of distance taken to find the socket's edge and learn a GMM and Q-EM policy separately from their demonstrations. In this way we can evaluate whether it is possible to learn a successful policy given a few bad demonstrations (15 training trajectories for each policy). Our second evaluation consists of using a noisy explorative Greedy policy as a teacher to gather demonstrations which can then be used to learn a new policy, which we call Q-Greedy.

Figure 4.17 illustrates 6 trajectories of Teacher # 5. The human teacher preferred to localise himself at the top of the wall before either proceeding to a corner or going directly towards the socket. Once localised, the teacher would reposition himself in front of the socket and try to achieve an insertion. This behaviour was not expected since by losing contact with the wall, the human teacher no longer had sensory feedback necessary to maintain an accurate position estimate.

Figure 4.18 illustrates the value function of the belief state learned from the data of teacher # 5. The states with the highest values seem to create a path going from the socket towards the right edge of the wall. We proceed as before to learn a GMM policy from the raw data and a Q-EM policy in which the data points are weighted by the gradient of the value function. In Figure 4.19, we illustrate the resulting Marginalised Gaussian Mixture parameters for both the GMM and Q-EM policies and we plot 25 rollouts of each policy starting at the *Center* initial condition used in Experiment 1. We note that the trajectories of the GMM policy have much variance in contrast to the Q-EM policy, resulting from an excess of variance in the 15 original demonstrations given by the teacher. Too much variance is not necessarily good, a random (uniform) policy in terms of generated trajectories will have the most variance and is as expected extremely inefficient in achieving a goal. Furthermore there is insufficient data to encode a pattern for the GMM model. In contrast, the Q-EM finds a pattern by combining multiple parts of the available data and as a result fewer data

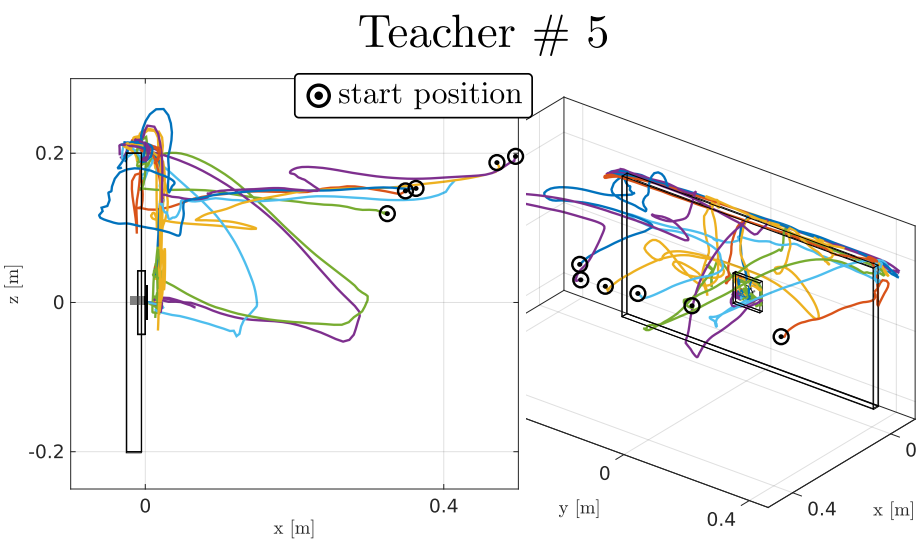


Figure 4.17: Demonstrations of Teacher # 5. The teacher demonstrates a preference

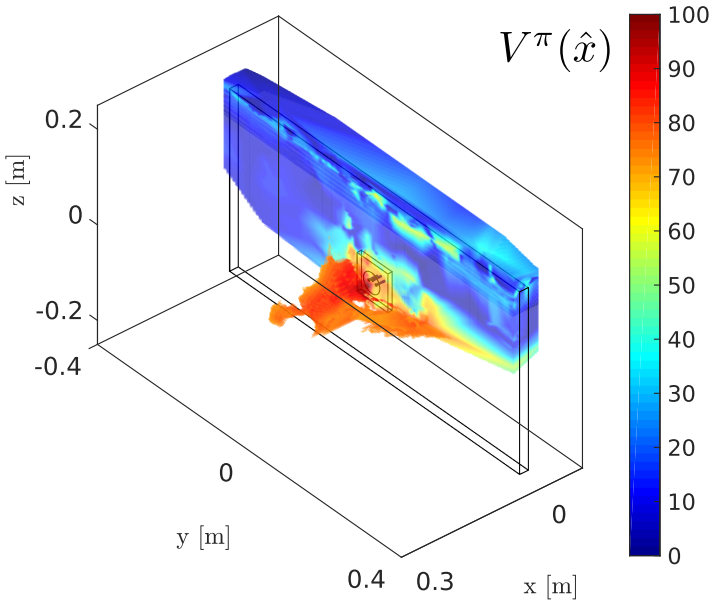


Figure 4.18: Value function learned from the 15 demonstrations of Teacher #5. The value of the most likely state is plotted.

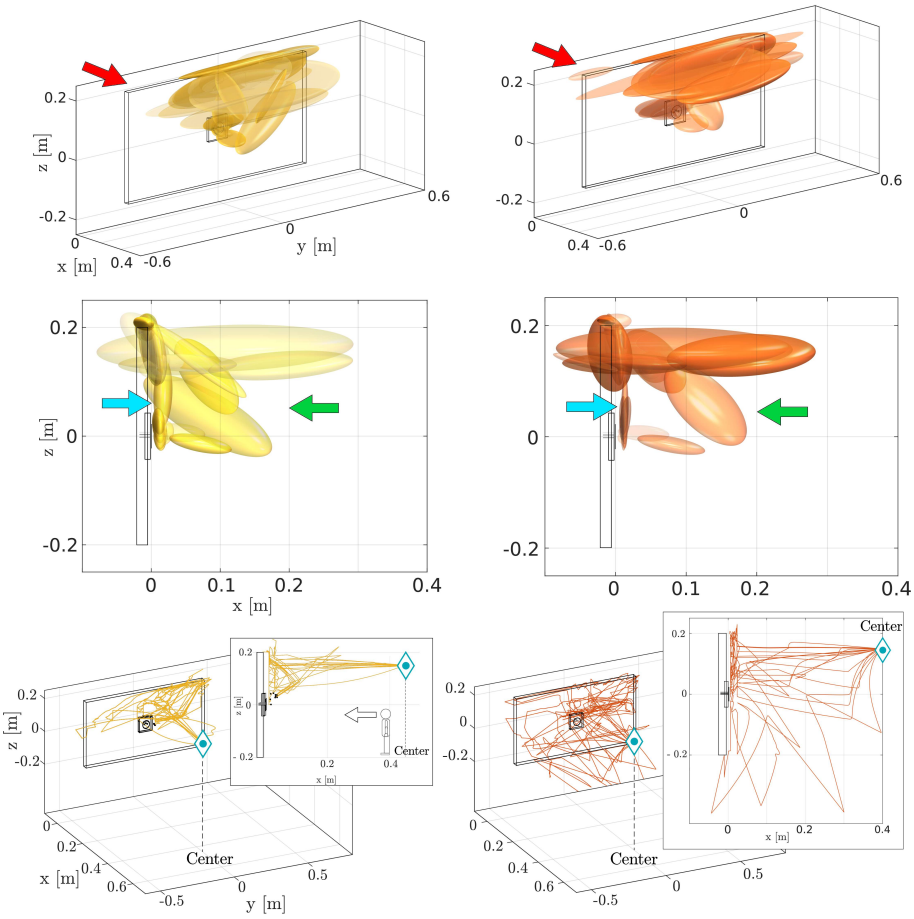


Figure 4.19: Marginalised Gaussian Mixture parameters of the GMM and Q-EM learned from the demonstrations of teacher #5. The illustrated transparency of the Gaussian functions is proportional to their weight. *Left column*: The Gaussian functions of the Q-EM have shifted from the left corner to the right. This is a result of the value function being higher in the top right corner region, see Figure 4.18. *Center column*: The original data of the teacher went quite far back which results in a Gaussian function given a direction which moves away from the wall (green arrow), whilst in the case of the Q-EM parameters this effect is reduced and moved closer towards the wall. We can also see from the two plots of the Q-EM parameters that they then follow the paths encoded by the value function. *Right column*: Rollouts of the policies learned from teacher #5. We can see that trajectories from the GMM policy have not really encoded a specific search patterns, whilst the Q-EM policy gives many more consistent trajectories which replicate to some extent the pattern of making a jump (no contact with the wall) from the top right corner to the socket's edge.

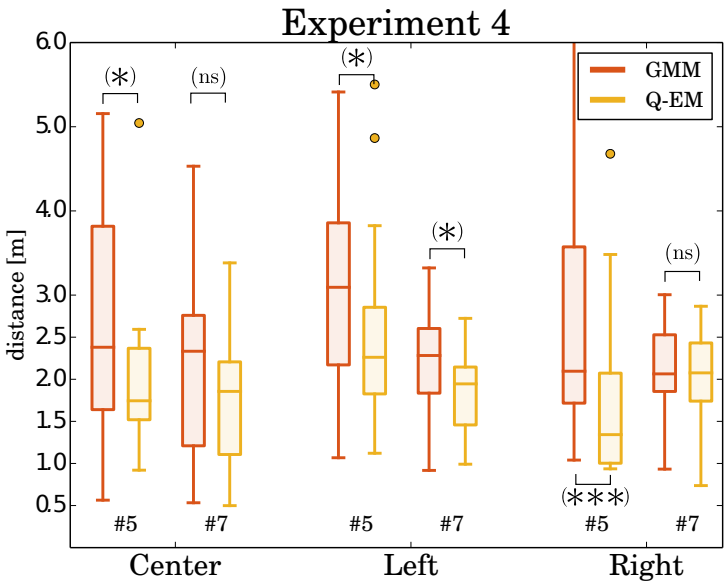


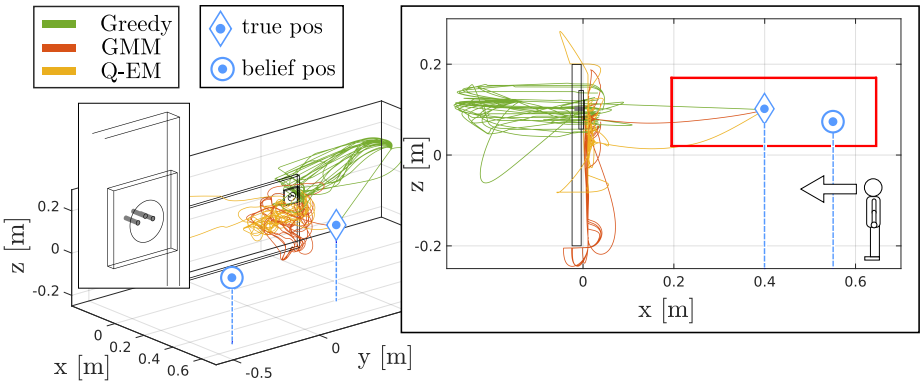
Figure 4.20: Results of a GMM and Q-EM policy under the same test conditions as Experiment 1. The Q-EM policy nearly always does much better than the GMM policy.

points are necessary to achieve a good policy. This effect is clear in Figure 4.20, showing the performance of the GMM and Q-EM algorithms under the same initial conditions as in Experiment 1. For all the conditions and for both teachers #5 and #7 the Q-EM policy always does better than the GMM.

We also tested whether we could use the Greedy policy as a means of gathering demonstrations in order to learn a value function and train a Q-Greedy policy. We used the Q-Greedy algorithm in combination with random perturbations applied to the Greedy velocity, to act as a simple exploration technique. We performed a maximum of 150 searches, which terminated once the socket was found and used these demonstrations to learn a value function and GMM policy which we refer to as Q-Greedy. Figure 4.15 illustrates the statistical results of the Q-Greedy policy for Experiment 1 and 3, showing that there is no difference between two policies. Our exploration method is probably too simplistic to discover meaningful search patterns and we could probably devise better search strategies which would result in a good policy. However we have shown that human behaviour encompasses both exploration and exploitation behaviour which can be used to learn a new policy through our RL-PbD-POMDP framework.

4.6.4 Generalisation

An important aspect of a policy or any machine learning methodology is to be able to generalise. So far we have trained and evaluated our policy within the same environment. To test whether our GMM policies can generalise to a new setting we changed the location of the socket to the upper right corner of



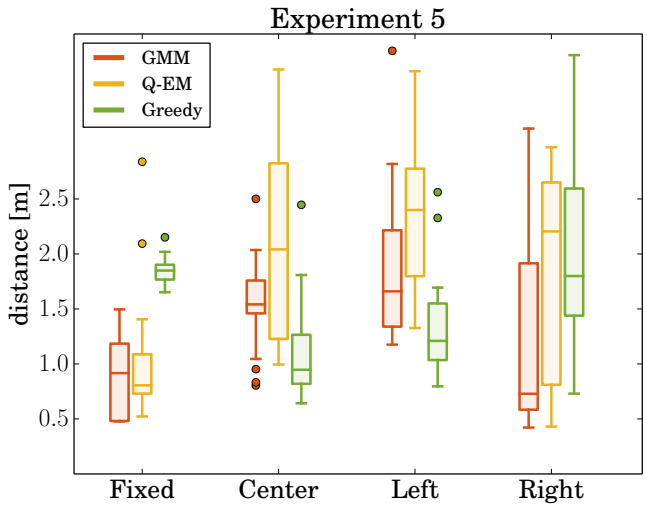
(a)

Figure 4.21: Evaluation of generalisation. The socket is located in at the top right corner of the wall. We consider a *Fixed* starting location for both the true and believed location of the end-effector. The red square depicts the extent of the initial uncertainty, which is uniform. (b) Distance taken to reach the socket's edge. For the Fixed setup (see (a) for the initial condition), both the Q-EM and GMM significantly outperform the Greedy. The other three conditions are the same as for Experiment 1.

the wall. The GMM was trained in the frame of reference of the socket and when we translated the socket's location it also translated the policy.

To evaluate the generalisation of our learned policy we use the same initial conditions of Experiment 1 with an additional new configuration named *Fixed*, in which both the true and believed location are fixed, blue triangle and circle. Figure 4.21 illustrates the trajectories of the three search policies for the *Fixed* initial condition. The Greedy policy moves in a straight line towards the top right corner of the table. As the true position is to the right, it takes the Greedy policy longer to find the wall in contrast to both the GMM and Q-EM policies. From the statistical results shown in Figure 4.22 we can see that for the *Fixed* and *Right* initial condition, which are similar, both GMM and Q-EM are better. However, for the *Center* and *Left* initial condition this is no longer the case. The Greedy method is better under this condition since the socket is close to informative features (it is located close to the edges of the wall). Once the end-effector has entered in contact with the wall the actions of the Greedy policy always result in a decrease of uncertainty, which was not the case when the socket was located in the center of wall. Thus in both the *Fixed* and *Right* initial condition the Greedy method does worse because it takes longer to find the wall.

The GMM based policies are still able to generalise under different socket locations. In general, as the socket's location is moved further from the original frame of reference in which it was learned, the higher is the likelihood that the search quality degrades. We chose the upper right corner since it is the furthest point from the origin and the GMM and Q-EM policies were still able to find the socket. We note that the policy will always be able to find the socket once it has localised itself. This can be seen from the vector field of the GMM policy



(a)

Figure 4.22: Distance taken to reach the socket's edge. For the Fixed setup (see Figure 4.21) for the initial condition), both the Q-EM and GMM significantly outperform the Greedy.

when the uncertainty is low, see Figure 4.10 on page 101. In this case the policy is a sink function with a single point attractor.

4.6.5 Distance taken to connect the plug to the socket

In this section we evaluate the distance taken for the policies and humans to establish a connection, after the socket has been found. We start measuring the distance from the point that the plug enters in contact with the socket's edge until the plug is connected to the socket. All the following evaluations are done on a KUKA LWR4 robot. The robot's end-effector is equipped with a plug holder on which is attached a force-torque sensor, the same holders used during the demonstration of the human teachers. In this way both the teacher and robot apprentice share the same sensory interface.

We chose to have the robot's end-effector located to the right of the socket and a belief spread uniformly along the z-axis. See Figure 4.24 for an illustration of the initial starting condition. This initial configuration was used to evaluate the search policies for the three different sockets, see Figure 4.2 on page 88 for an illustration of the sockets. The same initial configuration for the evaluation of the three sockets was kept in order to observe the generalisation properties of the policies. As a reminder we used only the training data from demonstrations acquired during the search with socket A. Socket B has a funnel which should make it easier to connect whilst socket C should be more difficult as it has no informative features on its surface.

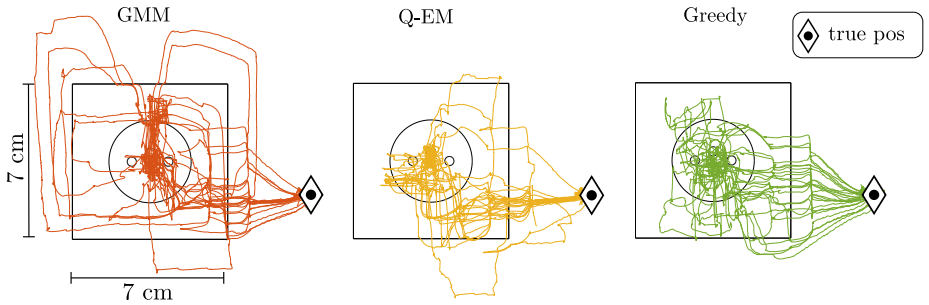


Figure 4.23: 25 search trajectories for each of the three search policies for socket A.

For each of the sockets we performed 25 searches starting from the same initial condition. In Figure 4.23 we plot the trajectories of each of the search methods for socket A. The GMM reproduces some of the behaviour exhibited by humans, such as first localising itself at the top of the socket before trying to attempt to make a connection. The Q-EM algorithm exhibits less variation than the GMM and tends to pass via the bottom of the socket to establish a connection. The Greedy method in contrast is much more stochastic since it does not take into consideration the variance of the uncertainty but tries instead to directly establish a connection. All three search methods are vastly superior, when compared to the human's performance see Figure 4.25. In Figure 4.24 illustrates a typical rollout of the GMM search policy for both socket A and C. Once a contact is made with the socket's edge the policy tends to stay close to informative features and tends to wander vertically up and down. Only when the uncertainty has been reduced does the GMM policy try to go towards the socket's connector.

The GMM and Q-EM policies are able to generalise to both socket B and C, as the geometric shape and connector interface of the two sockets are similar to socket A. The local force modulation of the policy's vector field, which is not learned, allows the end-effector to surmount edges and obstacles whilst trying to maintain a constant contact force in the x-axis. This modulation makes it possible for the plug to get on top of socket C.

Figure 4.26 illustrates the statistics of the distance taken to establish a connection for all three sockets. The interesting point is that both the GMM and Q-EM algorithms perform better than the Greedy approach for socket C. Socket C has no informative features on its surface and as a result myopic policies such as the Greedy policy will perform poorly. However for socket A and B, the Greedy policy performs better as both of these sockets have edges around their connector point allowing for easy localisation. It can also be seen that most search methods perform better on socket B than A, since the funnel shape connector helps in maintaining the plug within the vicinity of the socket's holes.

The discrepancy between the humans performance and the search policies can be attributed to many causes. One plausible reason is that the PMF probability density representation of the belief is more accurate than the human

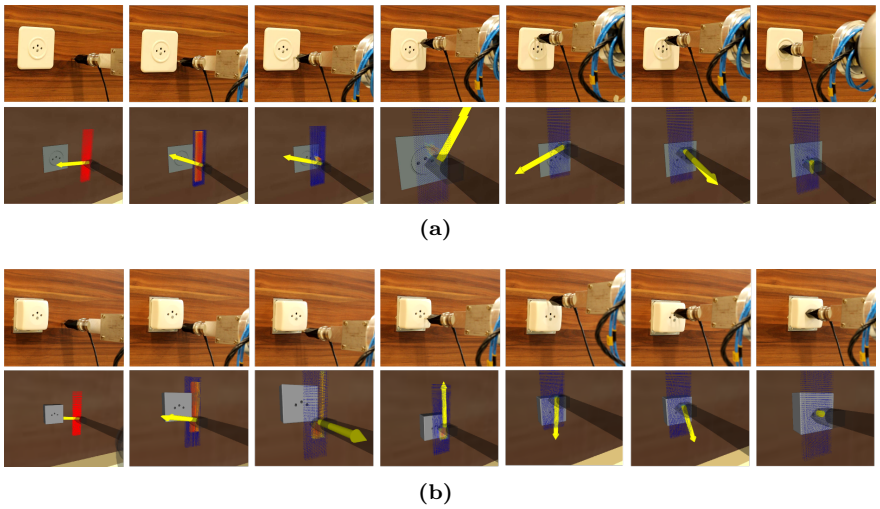


Figure 4.24: KUKA LWR4 equipped with a holder mounted with a ATI 6-axis force-torque sensor. (a) The robot's end-effector starts to the right of socket A. The second row shows screen captures taken of ROS Rviz data visualiser in which we see the Point Mass Filter (red particles) and a yellow arrow indicating the direction given by the policy. In this particular run, the plug remained in contact with the ring of the socket until the top was reached before making a connection. (b) Same initial condition as in (a) but with socket C. The policy leads the plug down to the bottom corner of the socket before going the center of the top edge, localising itself, and then making a connection.

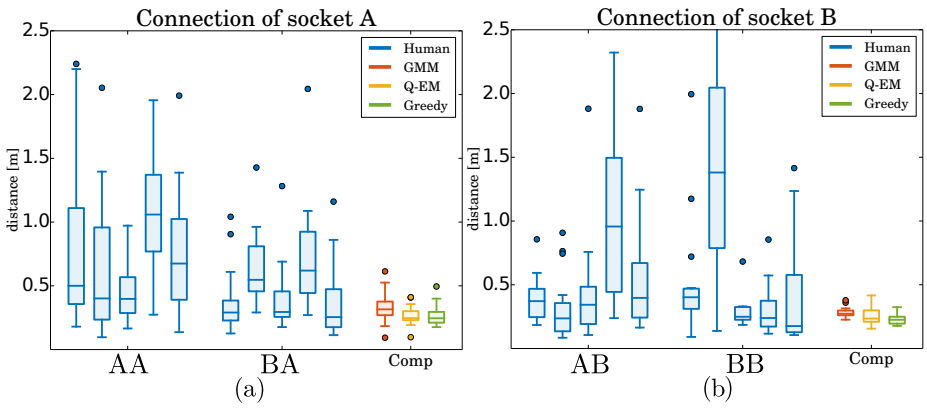


Figure 4.25: Distance taken to connect plug to socket once the socket is localised. (a) **Socket A**. The human Group A are the set of teachers who first started with socket A. They had no previous training on another socket beforehand. Group BA first gave demonstrations on Socket B before giving demonstrations on Socket A. Group BA is better than Group AA at doing the task. This is most likely a training effect. However all policy search methods are far better at connecting the plug to the socket. (b) **Socket B**. Both Groups AB and BB are similar in terms of the distance they took to insert the plug into the socket and as was the case for (a), the search policies travel less to accomplish the task.

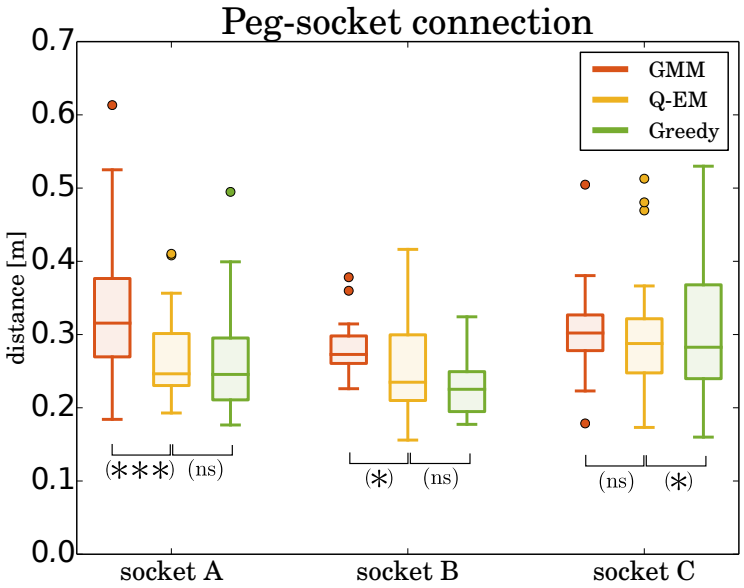


Figure 4.26: Distance taken (measured from point of contact of plug with socket edge) to connect the plug to the socket.

teachers position belief. Also, the motion noise parameter was fixed to be proportional to the velocity and the robot moves at gentle pace ($\sim$ 1 cm/s) as opposed to some of the human teachers. In actuality, humans are far less precise than the KUKA which has sub-millimetre accuracy.

4.7 Discussion and Conclusion

In this work we learned search policies from demonstrations provided by human teachers for a task which consisted of first localising a power socket (either socket A, B or C) and then connecting it with a plug. Only haptic information was available as the teachers were blindfolded. We made the assumption that the position belief of the human teachers was initially uniformly distributed in a fixed rectangular region of which they were informed and is considered prior knowledge. All subsequent beliefs were then updated in a Bayesian recursion using the measured velocity obtained from a vision tracking system, and wrench acquired from a force torque sensor attached to the plug. The filtered probability density function, represented by a Point Mass Filter, was then compressed to the most likely state and entropy.

Two Gaussian Mixture Model policies were learned from the data recorded during the human teachers' demonstrations. The first policy, called Q-EM, was learned in an Actor-Critic RL framework in which a value function was learned over the belief space. This was then used to weight training datapoints in the M-step update of Expectation-Maximisation (EM). The second policy, called GMM, was learned using the standard EM algorithm as in Chapter 3,

and considered all training data points equally, following in the footsteps of our initial approach de Chambrier and Billard (2014). Both the Q-EM and GMM policies were trained with data solely from the human demonstrations of the search with socket A.

Four different aspects of the learned policies have been evaluated. Firstly, which of three policies, Q-EM, GMM and a Greedy policy, took the least distance to find the socket. Across three different experiments it was shown that the Q-EM algorithm always performs the best. It was clear that the Q-EM policy was less random and more consistent than the GMM policy as it tried to enter in contact with the wall at the same height as the socket thus increasing the chances of finding the socket.

Secondly, the importance of the data provided by the human teachers was tested. The data from the two worst teachers was used to train an individual GMM and Q-EM policy for each of them. It was found that the performance of the Q-EM was better than the GMM in terms of distance travelled to find the socket. When qualitatively evaluating the trajectories of the GMM with respect to the Q-EM for the worst teacher, it is clear that the Q-EM policy managed to extract a search pattern, which was not the case for the GMM policy. A Q-EM policy was also learned from the data provided by a Greedy policy with explorative noise and no improvement was found. From these results we conclude that the exploration and exploitation aspects of the trajectories provided by the human teachers is necessary.

Thirdly, the two policies (GMM and Q-EM) were tested to see whether they were able to generalise to a different socket location. Under a specific condition, called *Fixed*, both policies were significantly better than the Greedy policy. However for the *Center* and *Left* initial conditions the Greedy policy performed better. When the Greedy policy enters in contact with the wall at an early stage, it performs better than the GMM and Q-EM. The reason for this is that the actions taken by the Greedy policy in this setting will always result in a decrease of entropy when the location of the socket is close to a corner, as opposed to being in the center of the wall. The reason this behaviour was not learned by the Q-EM approach is that no data showing this behaviour was present. One option would be to introduce an autonomous exploration mechanism such to discover this behaviour.

Fourthly, all three policies were evaluated on the KUKA LWR robot and all performed better than the human teachers. For socket A there is no clear distinction between the Q-EM and Greedy policy. On socket B, which was novel, the Greedy policy performed better than the statistical controllers, which we hypothesize was a result of a funnel which would make it easier for a myopic policy. For socket C, both the GMM and Q-EM policies performed better than the Greedy, as socket C has no features on its surface, this being a disadvantage for a policies which do not actively minimise uncertainty.

We conclude that by simply adding a binary reward function in combina-

tion with data provided by human demonstrations, using fitted reinforcement learning, better policy can be learned without the need to perform expensive exploration-exploitation rollouts traditionally associated with reinforcement learning and designing complicated reward functions. This is especially advantageous when only a few demonstrations are available.

Chapter 5

Non-parametric Bayesian State Space Estimator

In both Chapters 3 and 4, we demonstrated that it is feasible to learn a POMDP policy from human teachers. Further, by adding a simple binary reward function we were able to take into consideration the quality of these demonstrations provided by the teachers. With this, we showed that our Reinforcement Learning extension, RL-PbD-POMDP was able to yield improved policies even when provided with a limited number of demonstrations taken from the worst teachers.

Both tasks from the previous two chapters (search for wooden block on a table and peg-in-hole) fall into the category of goal oriented *active-localisation*. In general, the localisation problem consists of estimating position parameters given noisy observations whereas active-localisation refers to a policy which actively takes actions to acquire information to decrease the uncertainty of the position estimate. In localisation, the model of the world also known as the map is considered prior knowledge. This assumption constrains localisation to an environment in which schematics exist and can be used as the world model such as in the case of offices and buildings. If the map is not known a priori, then Simultaneous Localisation And Mapping (SLAM) algorithms have to be used instead of localisation, known as *active*-SLAM. Typically, the map consists of a set of features also known as landmarks which can be identified by sensors, and SLAM algorithms maintain a filtered joint probability distribution over both the agent's and features' position which is updated in accordance with a generic Bayesian State Space Filter (BSSF) (see Figure 2.5 on page 22).

In this chapter, we consider an agent tasked with searching for a set of objects on a *Table* world (see Figure 5.1), in which exteroceptive feedback is extremely limited. The agent can only sense an object after making physical contact with it (bumping into it). The agent's uncertainty of its location and that of the object are encoded by probability distributions $P(\cdot)$, which at initialisation are known as the agent's prior beliefs. Figure 5.1 illustrates a particular instance of the agent's beliefs. The agent is currently located in the bottom table and has only a limited knowledge of its location, somewhere near the right edge of the table.

As the agent explores the world, it integrates all sensing information at each time step and updates its prior beliefs to posteriors (the result of the prior belief

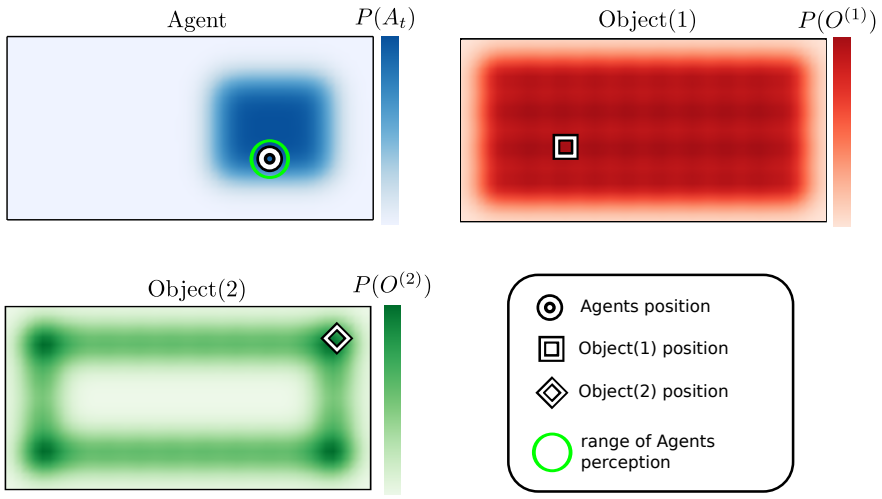


Figure 5.1: *Table Environment* Table World (delimited by the black rectangle), viewed from above, and the agent's beliefs. There are three different probability density functions present on the table. The blue represents the believed location of the agent, the red and green probability distributions are associated with object 1 and 2. The white shapes in each figure represent the true location of each associated object or agent.

after integrating motion and sensory information). In the search task illustrated in Figure 5.1, the beliefs and sparse measurement information (haptic) available to the agent are the source of the uncertainty which is, the absence of positive object measurements. In this setting as the sensory information is strictly haptic, we can confidently assume no measurement noise. This is known as *negative information* (Thrun et al., 2005, p.313) (Thrun, 2002; Hoffman et al., 2005). To the authors knoweldge, current SLAM methods are limited to non-negative information in that they consider only uncertainty induced by sensing inaccuracy inherent in the sensor and motion models (see the three main paradigms of SLAM (Thrun et al., 2005, Chap. 10-13)). Thus SLAM methodologies which use the *Gaussian error* between the predicted and estimated position of features, such as in the case of EKF-SLAM and Graph-SLAM, will not perform well in this setting.

The EKF SLAM algorithm, [...], can only process positive sightings of landmarks. It cannot process negative information that arises from the absence of landmarks. -Probabilistic Robotics (Thrun et al., 2005, p.313)-

In addition to the negative sensing information, the original beliefs depicted in Figure 5.1 are *non-Gaussian* and *multi-modal*. We make **no assumption** regarding the form of the beliefs. This implies that the joint distribution can no longer be parameterised by a Multivariate Gaussian. This is a necessary assumption for EKF-SLAM and its derived methods which allows for a closed form solution to the state estimation problem. Without the Gaussian assumption EKF-SLAM methods have no closed form solution to the filtering problem.

Attributes & Assumptions

- Non-Gaussian joint distribution, no assumptions are made with respect to its form.
- Mostly negative information available (absence of positive sightings of the landmarks).
- Joint distribution volume grows exponentially with respect to the number of objects and states.
- Joint distribution volume is dense, there is high uncertainty.

Figure 5.2: Assumptions and attributes which have to be fulfilled by our Bayesian State Space Filter.

Using standard non-parametric methods such as Kernel Density, Gaussian Process, Histogram to represent or estimate the joint distribution becomes unrealistic after a few dimensions or additional map features. FastSLAM could be a potential candidate, however as it parameterises the position uncertainty of the agent by a particle filter and each particle has its own copy of the map, the memory demands become quickly significant. For planning purposes we would also want to have a single representation of the marginals. Figure 5.2 summarises the desirable attributes and assumptions for our filter. This chapter is under review in de Chambrier and Billard (2016b).

The main contribution of our work and the importance to the field of Artificial Intelligence:

An accurate estimate of the agent's belief space is a necessary precondition before planning or reasoning can be carried out. In a wide range of Artificial Intelligence (AI) applications the agent's beliefs are discrete. This non-parametric representation is the most unconstraining but comes at a cost. The parameterisation of the belief's joint distribution grows at an exponential rate (see section 5.4.2 on page 141 for a details). We propose a Bayesian state estimator which delivers the same filtered beliefs as a traditional filter but without explicitly parametrising the joint distribution. Through memorising all the parameters of the measurement likelihood functions and by taking advantage of their sparsity (only a few states in the joint distribution are changed at any given time), we achieve a filter which grows linearly as opposed to exponentially in both time and space complexity. We refer to our novel filter as the Measurement Likelihood Memory Filter (MLMF). It keeps track of the history of measurement likelihood functions, referred to as the memory, which have been applied on the joint distribution. The MLMF filter efficiently processes negative information. To the best of the author's knowledge there has not been any research on negative information in an *active*-SLAM setting. By contrast there has been work considered with negative information in *active*-localisation (only agents

position is unknown, map is known)(Hoffmann et al., 2006). The incorporation of negative information is useful in many contexts especially in Bayesian Theory of Mind (Baker et al., 2011) where the reasoning process of a human is inferred from a Bayesian Network and in our own work de Chambrier and Billard (2014) where we model the search behaviour of an intentionally blinded human. In such a setting much negative information is present and an efficient belief filter is required. Our MLMF is thus applicable to the SLAM and AI community in general and to the cognitive community which models human or agent behaviours through the usage of Bayesian state estimators.

By using this new representation we implement a set of passive search trajectories through the state space and demonstrate, for a discretised state space, that our novel filter is optimal with respect to the Bayesian criteria (the successive filtered posteriors are exact and not an approximation with respect to Bayes rule). We provide an analysis of the space and time complexity of our algorithm and prove its efficiency even in the worst case scenario. Lastly we consider an Active-SLAM setting and evaluate how constraining the size of the number of memorised likelihood functions impacts the decision making process of a greedy one-step look-ahead planner.

5.1 Outline

The remaining part of this chapter is structured as follows:

- 5.2 Background: Review of three prominent SLAM algorithms and their assumptions and an overview of active-localisation and exploration methods used with SLAM.
- 5.3 Bayesian State Space Estimation: Introduction to EKF-SLAM and its unsuitability when mostly negative information is available. Description of the Histogram-SLAM algorithm and the assumptions which can be exploited.
- 5.4 Measurement Likelihood Memory Filter: Mathematical derivation of the MLMF, time and space complexity evaluation and extension to the scalable-MLMF.
- 5.5 Evaluation: We numerically evaluate the time complexity of the scalable-MLMF and verify its assumptions. We investigate the filter's sensitivity with respect to its parameters in an Active-SLAM setting.
- 5.6 Conclusion

5.2 Background

5.2.1 SLAM

Estimating the location or state parameters of a mobile agent whilst simultaneously building a map of the environment has been regarded as one of the most important problems to be solved for agents to achieve true autonomy. It is a necessary precondition for any agent to have an estimation of the world at its disposal which accurately encompasses all knowledge and related uncertainties. There has been much research surrounding the field of Simultaneous Localisation And Mapping (SLAM) which branches out into a wide variety of sub-fields dealing with problems from building accurate noise models of the agent sensors (Plagemann et al., 2007) to determining which environmental feature caused a particular measurement, also known as the data association problem (Montemerlo and Thrun, 2003) and many more.

Although the amount of research might seem overwhelming at first view, all current SLAM methodologies are founded on a single principle; the uncertainty of the map is correlated through the agent's measurements. When an agent localises itself (by reducing position uncertainty) all previously located landmarks have their uncertainty reduced since the uncertainty is correlated with that of the agent's uncertainty.

There are three main paradigms to solving the SLAM problem. The first is EKF-SLAM (Extendend-Kalman Filter) Durrant-Whyte and Bailey (2006). EKF-SLAM models the full state, being the agent's position parameters and environmental features, by a Multivariate Gaussian distribution. The uncertainty of each individual feature is parametrised by a mean (expected position of the feature) and covariance (the level of uncertainty of the position of the feature).

The second approach is Graph-SLAM, Grisetti et al. (2010). Graph-SLAM estimates the full path of the agent and considers every measurement to be a constraint on the agent's path. It is parameterised by the canonical Multivariate Gaussian. At each time step a new row and column is added to the precision matrix which encodes landmarks which have been observed as constraints on the robot's position. At predetermined times, a nonlinear sparse optimisation is solved to minimise all the accumulated constraints on the robot's path.

The third method is FastSLAM, Montemerlo et al. (2003). FastSLAM exploits the fact that if we know the agent's position with certainty all landmarks become independent. It models the distribution of the agent's position by a particle filter. Each particle has its own copy of the map and updates all landmarks independently which is the strength of this method. However, if many particles are required each must have its own copy of the map. It is beyond the scope of this chapter to provide a detailed review of these three paradigms and the reader is referred to Thrun et al. (2005), Thrun and Leonard (2008).

5.2.2 Active-SLAM & Exploration

Active-SLAM refers to a decision theoretic process of choosing control actions so as to actively increase the convergence of the map. It is used in conjunction with exploration of an unknown environment in a SLAM setting. The two steps of this process are: (i) generate a set of candidate destination positions, (ii) evaluate these positions based on a utility function. The utility is a trade off between reducing the uncertainty of the map or reducing the uncertainty of the agent's position.

Most approaches use a two-level representation of the map in an exploration setting. At the lower level there is the chosen (landmark-based) SLAM filter and at the higher level a coarser representation of the world. Such representations can be occupancy grids (Thrun and Bü, 1996) which encode either occupied and free space or a topological representation, Kollar and Roy (2008).

Early and current approaches to selecting candidate exploratory locations are based on evaluating *Next-best-view* locations, González-Baños and Latombe (2002). Next-best-view points are sampled around *free edges* which are at the horizon of the known map (*frontier* regions). In such a setting only target points are generated, not the full trajectory. Probabilistic Road Map (PRM) (Kavraki et al., 1996) methods have been used as planners to reach desired target locations, such as in Huang and Gupta (2008), where a Rapidly Exploring Random Trees (RRT) is combined with FastSLAM. In Valencia et al. (2012), paths to *frontier* regions are computed via PRM on a occupancy grid map and at the lower level they use Pose-SLAM (synonym for Graph-SLAM).

An alternative approach taken to generating candidate locations is the specification of high level macro actions, they being either *exploratory* or *revisiting* actions as is the case in Stachniss et al. (2005). Macro actions reduce the costly evaluation of actions, especially in the case of FastSLAM, which requires propagating the filter forward in time so as to infer the information gain of each action.

The last approach is to solve the planning problem through formulating it as Partially Observable Markov Decision Process (POMDP) (Ross et al., 2008). However all methods take an approximation of the POMDP and consider a one time step planning horizon (Lidoris, 2011, p.37).

There are many ways of generating actions or paths, however their utility is nearly all exclusively based on the *information gain*, which is the estimated reduction of entropy a particular action or path would achieve. A few utilities use f-measures such as the Kullback-Leibler divergence. Evaluation of different utility metrics are presented in Carrillo et al. (2012).

5.3 Bayesian State Space Estimation

Bayesian State Space Estimation (BSSE) focuses on incorporating observa-

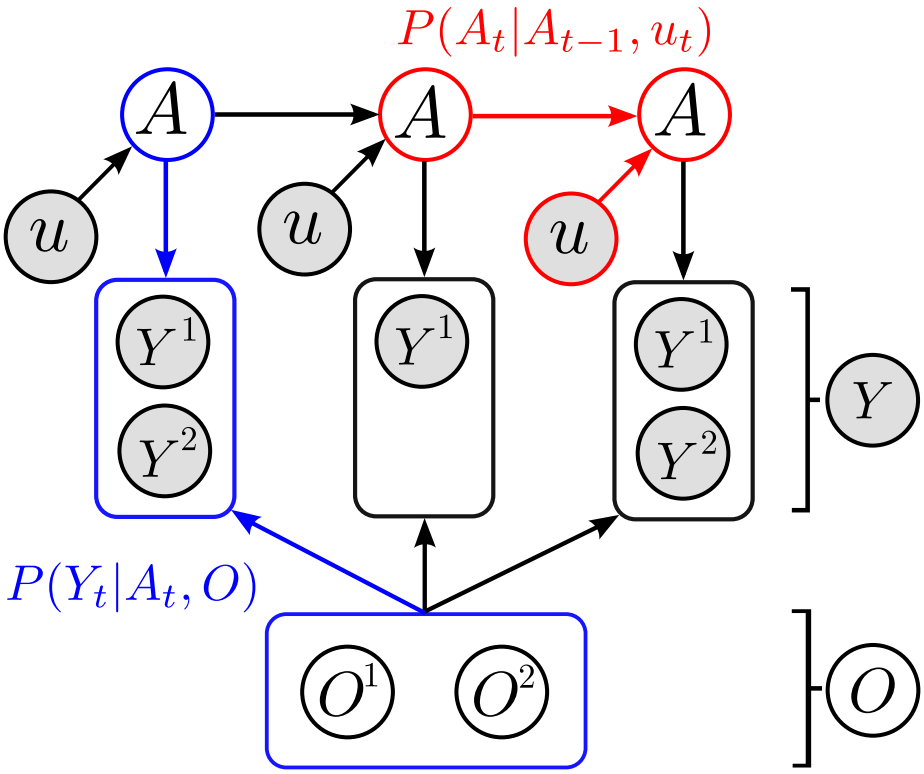


Figure 5.3: Directed graphical model of dependencies between the agent(A) and object(O)'s estimated location. Each object, $O^{(i)}$ is associated with one sensing random variable $Y^{(i)}$. The overall sensing random variable is $Y = \left[Y^{(1)}, \ldots, Y^{(M-1)}\right]^{\mathrm{T}}$, where M is the total number of agent and object random variables in the filter. For readability we have left out the time index t from A and Y. Since the objects are static, they have no temporal process associated with them thus they will never have a time subscript. The two models necessary for filtering are the motion model $P(A_t|A_{t-1}, u_t)$ (red) and measurement model $P(Y_t|A_t, O)$ (blue).

tions to update a prior distribution to a posterior distribution over the state space through the application of Bayes probability rules. The agent's random variable, A, is associated with the uncertainty of its location in the world. The same holds for the object(s') random variable(s), O. Given a sequence of actions and observations, $\{u_{1:t}, Y_{0:t}\}$ (subscript $0:t$ is all the indexed variables from $t = 0$ to the current time $t = t$), algorithms of the BSSE family incorporate this information to provide an estimate $P(A_t, O|Y_{0:t}, u_{1:t})$. This is known as the filtering problem where all past information is incorporated to estimate the current state.

In Figure 5.3 we depict the general Bayesian Network (BN) of a BSSE. The BN conveys two types of information, the dependence and independence relation between the random variables in the graph which can be established through *d-separation* Shachter (1998). The **conditional dependence** $A \top O|Y$ is key to all BSSE and SLAM algorithms. The strength of the dependence between the agent and object random variable is governed by the measurement likelihood $P(Y_t|A_t, O)$. If the measurement likelihood does not change the joint

distribution, then the agent and object random variables will be independent, $A \perp\!\!\!\perp O$. If they are independent, then no information acquired by the agent can be used to infer changes in the object estimates.

We next demonstrate the behaviour of the BN joint distribution, Figure 5.3, for two different parameterisations in the case of the absence of direct sighting of the object by the agent.

EKF-SLAM

In EKF-SLAM the joint density $p(A_t, O|Y_{0:t}, u_{1:t}) = g([A_t, O]^{\mathrm{T}}; \mu_t, \Sigma_t)$ is parametrised by a single Gaussian function $g(\cdot)$ with mean, $\mu_t = [\mu_{A_t}, \mu_{O^{(1)}}, \ldots, \mu_{O^{(M-1)}}]^{\mathrm{T}} \in \mathbb{R}^{3+2\cdot(M-1)}$ where the object random variables are in $\mathbb{R}^2$, and covariance, Σ_t. The mean value of the agent $\mu_a = [x, y, \phi]^{\mathrm{T}} \in \mathbb{R}^3$ and those of the objects are $\mu_{O^{(i)}} = [x, y]^{\mathrm{T}} \in \mathbb{R}^2$.

$$\Sigma_t = \begin{bmatrix} \Sigma_a & \Sigma_{ao} \\ \Sigma_{oa} & \Sigma_o \end{bmatrix} \in \mathbb{R}^{(3+2\cdot(M-1))\times(3+2\cdot(M-1))} \tag{5.3.1}$$

The j'th object measurement is described by range and bearing $Y_t^{(j)} = [r, \phi]$ in the frame of reference of the agent. EKF-SLAM assumes that the measurement is corrupted by Gaussian noise, $\epsilon \sim \mathcal{N}(0, R)$, resulting in the likelihood function:

$$p(Y_t|A_t, O_t) = \frac{1}{|2\pi R|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(Y_t - \hat{Y}_t\right)^{\mathrm{T}} R^{-1}\left(Y_t - \hat{Y}_t\right)\right) \tag{5.3.2}$$

$$\hat{Y}_t = \exp\left(-\frac{1}{2\sigma^2}||A_t - O||^2\right) \tag{5.3.3}$$

where the covariance, R, encompasses the uncertainty in the measurement and Equation 5.3.3 is the measurement function. The elements of the covariance matrix capture the measurement error between the true Y and expected $\hat{Y}$ range and bearing of the object. As the joint distribution is parametrised by a single Multivariate Gaussian, a closed form solution to the filtering Equations exists, called the Kalman Filter Durrant-Whyte and Bailey (2006).

The error between the true and expected measurement $e = (Y_t - \hat{Y}_t)$ is an important part of the application of EKF-SLAM. In our scenario the agent can only perceive the objects once he enters in direct contact with them. This means that the variance of the observation Y_t will always be equal to $\hat{Y}$ until a contact occurs. To illustrate the problems which this gives rise to, we give an illustration of a 1D search. Figure 5.4 shows the resulting updates of the beliefs for 4 chosen time segments.

As expected we do not get the desired behaviour, that the beliefs start updating as soon as they are overlapping, see 2nd-3rd temporal snapshot in the Figure. Even when most of the belief mass of the agent's location pdf overlaps

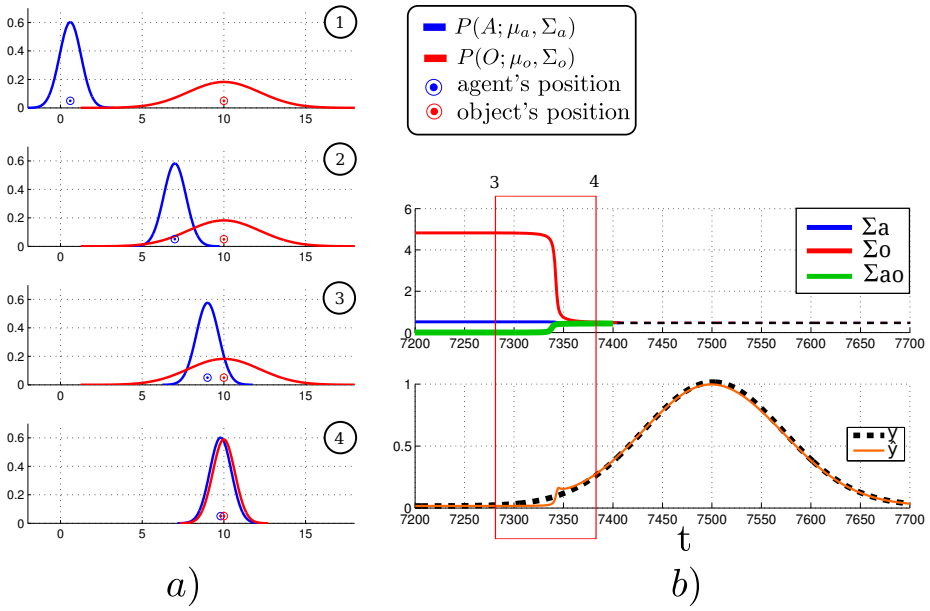


Figure 5.4: **a)** EKF-SLAM time slice evolutions of the pdfs. The temporal ordering is given by the numbers in the top right corner of each plot. The blue pdf represents the agent's believed location and the circle is the agent's true location. The same holds for the red distribution which represents the agent's belief of the location of an object. **b)** Evolution of the covariance components of Σ over time and true Y_t and expected measurements, $\hat{Y}_t$. Σ_a and Σ_o are the variances of the agent and object positions and Σ_{ao} is the cross-covariance term.

that of the object pdf, no belief update occurs. The multivariate Gaussian parameterisation only guarantees a dependency between the agent and object random variables when there is a positive sighting of the landmarks. This can been seen in Figure 5.4 (b), where the component Σ_{ao} is 0 most of the time which implies that $A \perp\!\!\!\perp O|Y$ which is undesirable.

Histogram-SLAM

In Histogram-SLAM, the joint distribution is discretized and each bin has a parameter, $P(A_t = i, O = j|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}) = \boldsymbol{\theta}^{(ij)}$, which sums to one, $\sum_{ij} \boldsymbol{\theta}^{(ij)} = 1$. For shorthand notation we will write $P(A_t, O|Y_{0:t}, u_{1:t})$ instead of $P(A_t = i, O = j|Y_{0:t}, u_{1:t}; \boldsymbol{\theta})$. The probability distribution of the agent's position is given by marginalising the object random variable:

$$P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a) = \sum_{j=1}^{|O|} P(A_t, O = j|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}) \tag{5.3.4}$$

The converse holds true for the object's marginal, that is the summation would be over the agents variable. Figure 5.5 (*Top*) illustrates the joint distribution of both the agent and the object random variable. The 1D world of the agent and object is discretised to 10 states, producing a joint distribution with

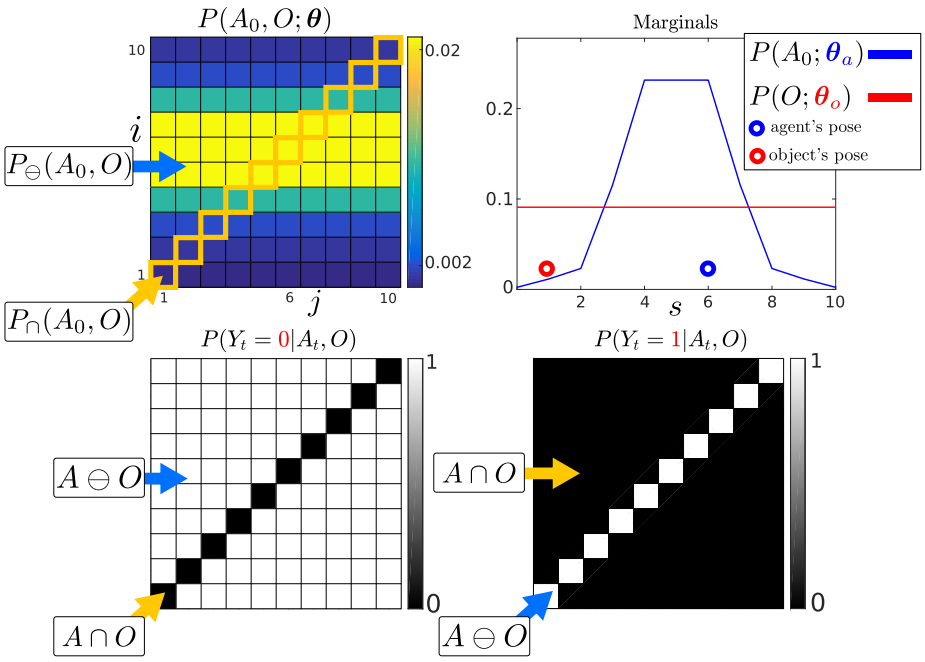


Figure 5.5: **Top**: *Left:* Initialisation of the agent and object joint distribution. *Right:* Marginals of the agent and object parameterised by $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_o$, giving the probability of their location. The marginal of each random variable is obtained from Equation 5.3.4. The probability of the agent and object being in state $s = 6$ is given by summing the blue and red highlighted parameters in the joint distribution. **Bottom**: 1D world Likelihood $P(Y_t|A_t, O)$, the white regions $A \cap O$ will leave the joint distribution unchanged whilst the black regions will evaluate the joint distribution to zero. *Left:* No contact detected with the object, the current measurement is $Y_t = 0$, both the agent and object cannot be in the same state. *Right:* The agent entered into contact with the object and received a haptic feedback $Y_t = 1$. The agent receives only two measurement possibilities, contact or no contact.

100 parameters! For a state space of N bins, $s = 1...N$, and there is a total of M random variables (one agent and $M - 1$ objects) and the joint distribution has N^M parameters. This exponential increase renders Histogram-SLAM intractable with this parameterisation.

In the tasks we consider, an observation occurs only if the agent enters in contact with the object, which implies that both occupy the same discrete state. The likelihood function $P(Y_t|A_t, O)$ is:

$$P(Y_t = 1|A_t, O) = \begin{cases} 1 & \text{if } A_t = O \\ 0 & \text{if } A_t \neq O \end{cases} \tag{5.3.5}$$

Figure 5.5 (*Bottom left*), illustrates the likelihood function, Equation 5.3.5, in the case of a no contact measurement $Y_t = 0$. When there is no measurement all the parameters of the joint distribution which are in the black regions become zero, which we refer to as the **dependent states** $A \cap O$ of the joint distribution. The white states are the **independent states** $A \ominus O$, they are not changed by the likelihood function and the values of the joint distribution in

those states, $P_{\cap}(A_t, O|Y_{0:t}, u_{1:t})$, will be unchanged by the likelihood function $P_{\ominus}(A_t, O|Y_{\mathbf{0:t}}, u_{1:t}) \propto P_{\ominus}(A_t, O|Y_{\mathbf{0:t-1}}, u_{1:t})$. When the object is detected (*Bottom right*) the likelihood constrains all non-zero values of the joint distribution to be in states $i = j$, which in the case of a 2-dimensional joint distribution is a line. The **sparsity** of the likelihood function will be key to the development of the MLMF filter. Two models are needed to perform the recursion, namely the motion model $P(A_t|A_{t-1}, u_t)$ and the measurement model $P(Y_t|A_t, O)$, which we already detailed. Both models applied consecutively to the initial joint distribution results in a posterior distribution. Both Equation 5.3.7-5.3.8 are part of the Histogram Bayesian filter update:

Histogram Bayesian recursion

intialisation

$$P(A_0, O; \boldsymbol{\theta}) = P(A_0; \boldsymbol{\theta}_a)\, P(O; \boldsymbol{\theta}_o) = \boldsymbol{\theta}_a \times \boldsymbol{\theta}_o \tag{5.3.6}$$

motion

$$P(A_t, O|Y_{0:t-1}, u_{1:t}) = \sum_{A_{t-1}} P(A_t|A_{t-1}, u_t)\, P(A_{t-1}, O_t|Y_{0:t-1}, u_{1:t-1}) \tag{5.3.7}$$

measurement

$$P(A_t, O|Y_{0:t}, u_{1:t}) = \frac{P(Y_t|A_t, O)\, P(A_t, O|Y_{0:t-1}, u_{1:t})}{P(Y_t|Y_{0:t-1}, u_{1:t})} \tag{5.3.8}$$

Figure 5.6 illustrates the evolution of the joint distribution in a 1D example. The agent and object's true positions (unobservable) are in state 6 and 1. The agent moves three steps towards state 10. At each time step, as the agent fails to sense the object, the likelihood function $P(Y_t = 0|A_t, O)$ (Figure 5.5, *Bottom left*) is applied. As the agent moves towards the right, the motion model shifts the joint distribution towards state 10 along the agent's dimension, i (note that state 1 and 10 are wrapped).

As the agent moves to the right more joint distribution parameters become zero. The re-normalisation by the **evidence** ($P(Y_t|Y_{0:t-1}, u_{1:t})$, denominator of Equation 5.3.8), which increases the value of the remaining parameters, is equal to the sum of the probability mass which was set to zero by the likelihood function. Thus the values of the parameters of the joint distribution which fall on the pink line in Figure 5.6 (green line also, but only for first time slice) become zero and their values are redistributed to the remaining non-zero parameters.

The **inconvenience** with Histogram-SLAM is that its time and space complexity is exponential as the joint distribution is discretised and parametrised by $\boldsymbol{\theta}^{(ij)}$. Instead we propose a new filter, MLMF, which we formally introduce in the next section. This filter achieves the same result as the Histogram filter but without having to parameterise the values of the joint distribution, thus avoiding the exponential growth cost.

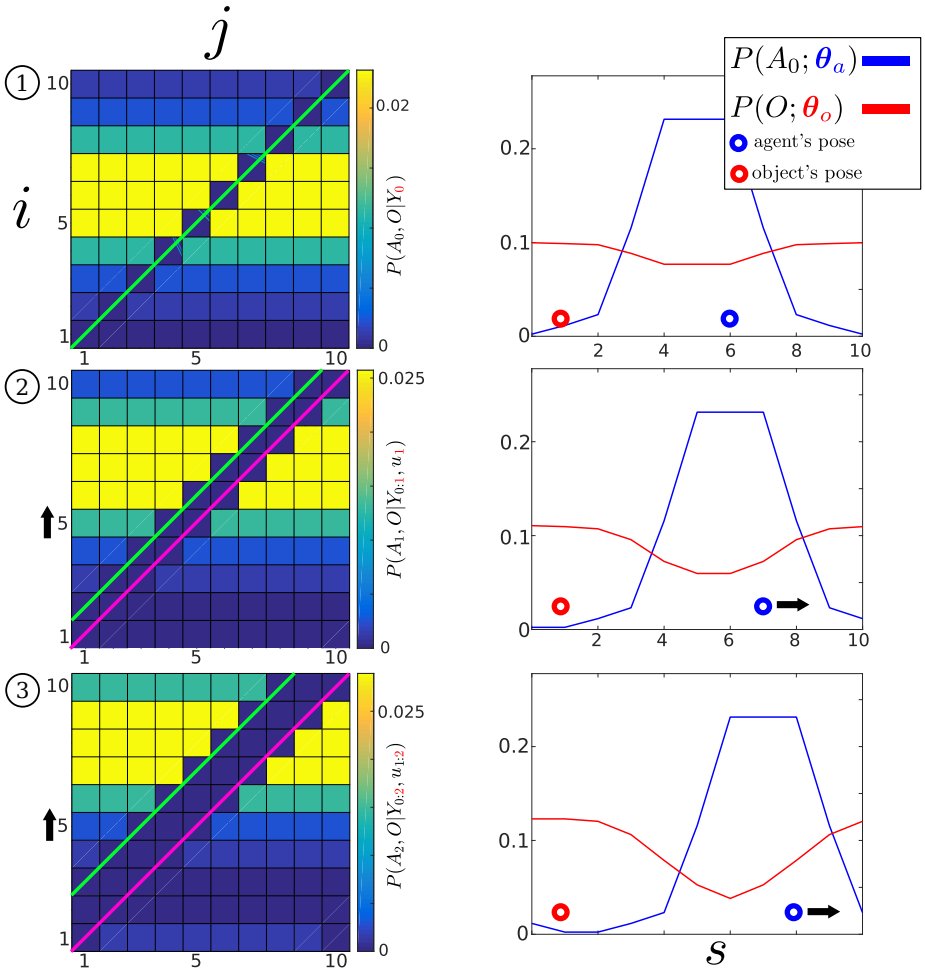


Figure 5.6: Histogram-SLAM, 3 time steps. **1** Application of likelihood $P(Y_0 = 0|A_0, O)$ and the agent remains stationary, all states along the green line become zero. **2** The agent moves to the right $u_1 = 1$, the motion $P(A_1|A_0, u_1)$, and likelihood models are applied consecutively. The right motion results in a shift (black arrow on the left) in the joint probability distribution towards the state $i = 10$. All parameters on the pink line are zero. **3** Same as two. At each time step a new likelihood function (pink line) is applied to the joint distribution.

The **key idea** behind the mechanism of the MLMF filter is to evaluate only the joint distribution $P_{\cap}(A_t, O|Y_{0:t}, u_{1:t})$ in dependent states and updates directly the marginals without marginalising the entire joint state space. The MLMF filter parametrises **explicitly** the marginals $P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_o)$. This contrasts the Histogram filter where the marginals are derived from the joint distribution by marginalisation over the entire joint state space.

5.4 Measurement Likelihood Memory Filter

MLMF keeps a **function parameterisation** of the joint distribution instead of a **value parameterisation** as it is the case for Histogram-SLAM. At initialisation the joint distribution is represented by the product of marginals, Equation 5.4.1, which would result in the joint distribution illustrated in Figure 5.5, if it were to be evaluated at all states (i, j) as it is done for Histogram-SLAM. MLMF will only evaluate this product, when necessary, at specific states. At each time step the motion and measurement update are applied, Equation 5.4.2-5.4.3. An important distinction is that these updates are performed on the **un-normalised** joint distribution $P(A_t, O, Y_{0:t}|u_{1:t})$, which is not the case in Histogram-SLAM where the updates are done on the conditional $P(A_t, O|Y_{0:t}, u_{1:t})$. After applying multiple motion and measurement updates the resulting joint distribution is given by Equation 5.4.4, see Appendix B.3 for a step-by-step derivation.

MLMF Bayesian filter

joint marginals (initial)

$$P(A_0, O) = P(A_0; \boldsymbol{\theta}_a^*)\, P(O; \boldsymbol{\theta}_o^*) \tag{5.4.1}$$

motion

$$P(A_t, O, Y_{0:t-1} | u_{1:t}) = \sum_{A_{t-1}} P(A_t | A_{t-1}, u_t)\, P(A_{t-1}, O, Y_{0:t-1} | u_{1:t-1}) \tag{5.4.2}$$

measurement

$$P(A_t, O, Y_{0:t} | u_{1:t}; \boldsymbol{\theta}_o^*, \boldsymbol{\theta}_a^*, \boldsymbol{\Psi}_{0:t}) =$$

$$P(Y_t | A_t, O)\, P(O; \boldsymbol{\theta}_o^*)\, P(A_t | u_{1:t}; \boldsymbol{\theta}_a^*)\, P(Y_{0:t} | A_t, O, u_{1:t}; \bar{\boldsymbol{\Psi}}_{0:t}) \tag{5.4.3}$$

joint

$$P(A_t, O | Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_o^*, \boldsymbol{\theta}_a^*, \boldsymbol{\Psi}_{0:t}, \boldsymbol{\alpha}_{0:t}) = \frac{P(A_t, O, Y_{0:t} | u_{1:t}; \boldsymbol{\theta}_o^*, \boldsymbol{\theta}_a^*, \boldsymbol{\Psi}_{0:t})}{P(Y_{0:t} | u_{1:t}; \boldsymbol{\alpha}_{0:t})} \tag{5.4.4}$$

filtered marginal

$$P(A_t | Y_{0:\mathbf{t}}; \boldsymbol{\theta}_a) = P(A_t | Y_{0:\mathbf{t-1}}; \boldsymbol{\theta}_a) - \Big(P_\cap(A_t | Y_{0:t-1}) - P_\cap(A_t | Y_{0:t})\Big) \tag{5.4.5}$$

$$P(O | Y_{0:\mathbf{t}}; \boldsymbol{\theta}_o) = P(O | Y_{0:\mathbf{t-1}}; \boldsymbol{\theta}_o) - \Big(P_\cap(A_t | Y_{0:t-1}) - P_\cap(A_t | Y_{0:t})\Big) \tag{5.4.6}$$

The MLFM filter is parameterised by the agent and object **joint marginals** $P(A_t | u_{1:t}; \boldsymbol{\theta}_a^*)$, $P(O; \boldsymbol{\theta}_o^*)$, the **filtered marginals** $P(A_t | Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$ ($u_{1:t}$ not shown in the above box), $P(O | Y_{0:\mathbf{t}}; \boldsymbol{\theta}_o)$, the evidence $P(Y_{0:t} | u_{1:t}; \boldsymbol{\alpha}_{0:t})$ and the history of likelihood functions, $P(Y_{0:t} | A_t, O, u_{1:t}; \boldsymbol{\Psi}_{0:t})$ Equation 5.4.7, which is the product of all the likelihood functions since $t = 0$ until t and we will refer to it as the **memory likelihood function**:

$$P(Y_{0:t} | A_t, O, u_{1:t}; \boldsymbol{\Psi}_{0:t}) := \prod_{i=0}^{t} P(Y_i | A_t, O, u_{i+1:t}; l_i) \tag{5.4.7}$$

$$P(Y_i = 0 | A_t, O, u_{i+1:t}; l_i) := \begin{cases} 0 & \text{if } A_t + l_i = O \\ 1 & \text{else} \end{cases} \tag{5.4.8}$$

$$l_i := \sum_{j=i+1}^{t} u_j \tag{5.4.9}$$

The memory likelihood function's parameters $\boldsymbol{\Psi}_{0:t} = \{(Y_i, l_i)\}_{i=0:t}$ consist of a set of measurements $Y_{0:t}$ and offsets $l_{0:t}$ depicted in green. The measurements $Y_i \in \{0, 1\}$ are always binary, whilst the offsets l_i, actions u_t, agent A_t and object O variables' size are equal to the dimension of the state space. The subscript i of an offset l_i indicates which likelihood function it belongs to. The offset of a likelihood function is given by the summation of all the applied actions from the time the likelihood was added until the current time t, Equation 5.4.9, which can be computed recursively. The motion update, Equation 5.4.2, when

applied to the joint distribution results in the initial marginal $P(A_0; \boldsymbol{\theta}^*_a)$ and the likelihood functions being moved along the agent's axis. In Algorithm 2, we detail how an action u_t and measurement Y_t, result in the update of the memory likelihood's parameters from $\Psi_{0:t-t}$ to $\Psi_{0:t}$; this is an implementation of Equations 5.4.2-5.4.3.

Algorithm 2: Memory Likelihood update

input : $\Psi_{0:t-1}$, Y_t, u_t

output: $\Psi_{0:t}$

---

motion update $\bar{\Psi}_{0:t} \leftarrow \Psi_{0:t-1}$

1 **for** $l_i \in \Psi_{0:t-1}$ **do**

2 $\quad l_i = l_i + u_t$

---

measurement update

3 $\Psi_{0:t} \leftarrow \{\bar{\Psi}_{0:t}, (Y_t, l_t := 0)\}$

Figure 5.7 illustrates the evolution of the **un-normalised** MLMF joint distribution, Equation 5.4.4. For ease of notation we will omit at times the parameters of the probability functions. Both $P(A_0; \boldsymbol{\theta}^*_a)$ and $P(O; \boldsymbol{\theta}^*_o)$ were initialised as for the Histogram-SLAM example in Figure 5.6 on page 132. Two actions $u_{1:2} = 1$ are applied and three measurements $Y_{0:2} = 0$ received which are then integrated into the filter. Since initialisation of the joint distribution at $t = 0$ until $t = 2$ the object's marginal $P(O; \boldsymbol{\theta}^*_o)$ remains unchanged and the agent's marginal $P(A_2|u_{1:2}; \boldsymbol{\theta}^*_a)$ is updated by the two actions according to the motion update, see Figure 5.7 *Top-right*. The product of these two marginals (terms of Equation 5.4.4 before the memory likelihood product) results in the joint probability distribution $P(A_2, O|u_{1:2}; \boldsymbol{\theta}^*_a, \boldsymbol{\theta}^*_o)$ illustrated in Figure 5.7 *Middle-right*.

In the left column of Figure 5.7 we illustrate how the memory likelihood term, Equation 5.4.7, is updated according to Algorithm 2. In the *Top-left*, the first likelihood function $P(Y_0|A_2, O, u_{1:2}; l_0)$ is illustrated. As two actions have been applied, Algorithm 2 is applied twice which results in a $l_0 = 2$ parameter for the first likelihood function. In the figure we highlighted the likelihood in light-green to indicate that it was the first added to the memory term making it convenient to compare to Figure 5.6 on page 132. As for the second likelihood function $P(Y_1|A_2, O, u_2; l_1)$ only one action has been applied and the third likelihood function $P(Y_2|A_2, O; l_2 = 0)$ has not yet been updated by the next action. The parameters of the memory likelihood function, Equation 5.4.7, are: $\Psi_{0:2} = \{(0, 2)_{i=0}, (0, 1)_{i=1}, (0, 0)_{i=2}\}$ and the evaluation of memory likelihood is depicted in the *Bottom-left* of Figure 5.7.

The reader may have noticed that the amplitude of the values of the joint distribution illustrated in Figure 5.7 have not changed when compared with Figure 5.6 on page 132. This is because we have not re-normalised the joint distribution by the evidence $P(Y_{0:t}|u_{1:t}; \alpha_{0:t})$.

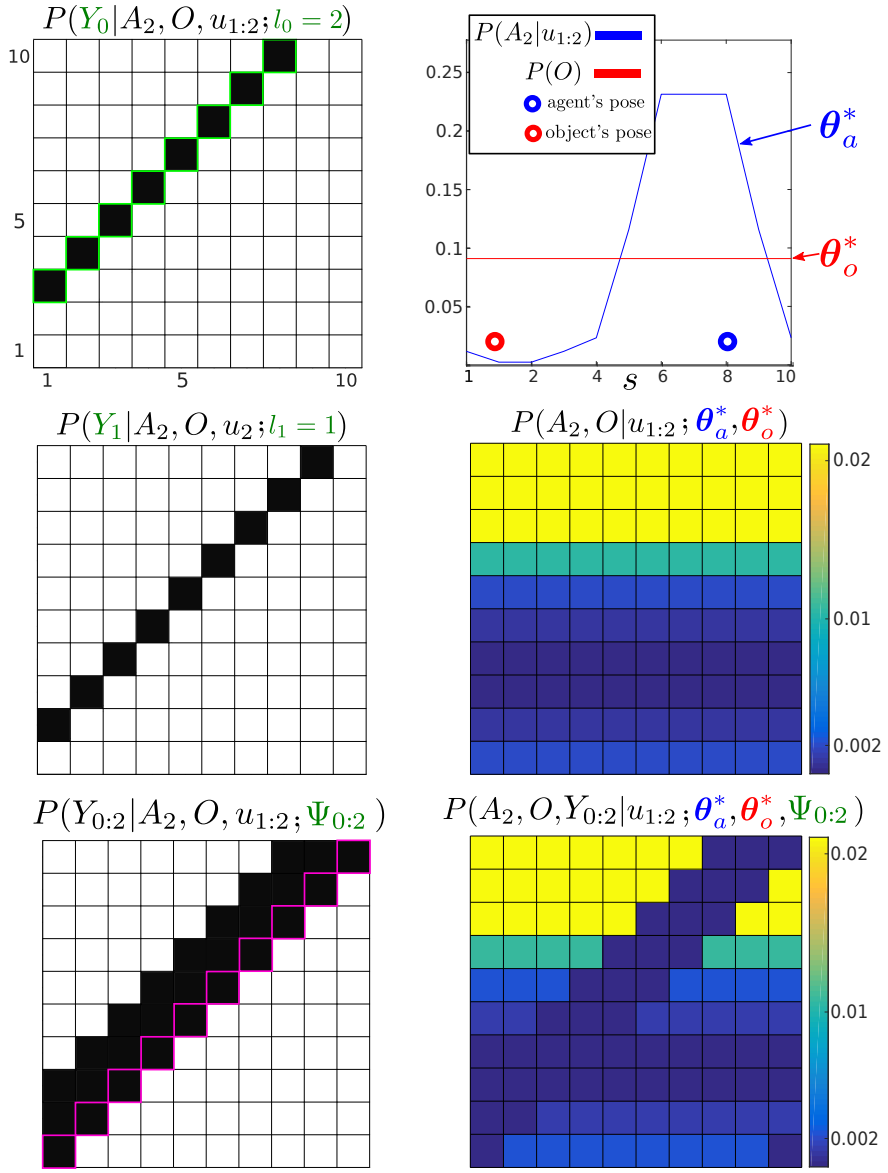


Figure 5.7: Un-normalised MLMF joint distribution, numerator of Equation 5.4.4, at time $t = 2$. Three measurements (all $Y = 0$) and two actions (both $u = 1$) have been integrated into the joint distribution, for simplicity we do not consider any motion noise. *Left column:* The first plot illustrates the likelihood of the first measurement Y_0. We highlight the contour in light-green to indicate that it was the first applied likelihood function (see the correspondence with Figure 5.6). The first likelihood function has been moved by the 2 actions, the second likelihood function has been moved by one action (the last one, $u_2 = 1$) and the third likelihood has had no action applied to it yet. The last applied likelihood function is highlighted in pink and the product of all the likelihoods since $t = 0$ until $t = 3$ is depicted at the bottom of the figure which is $P(Y_{0:2}|A_2, O, u_{1:2})$. *Right column:* the top figure illustrates the original marginal of the object $P(O; \boldsymbol{\theta}^*_o)$, which remains unchanged, and the agent's marginal $P(A_2|u_{1:2}; \boldsymbol{\theta}^*_a)$ which has moved in accordance to the motion update function. Their product would results in the joint distribution $P(A_2, O|u_{1:2}; \boldsymbol{\theta}^*_a, \boldsymbol{\theta}^*_o)$ illustrated in the middle figure if evaluated at each state (i, j). The bottom figure is the result of multiplying $P(A_2, O|u_{1:2}; \boldsymbol{\theta}^*_a, \boldsymbol{\theta}^*_o)$ with $P(Y_{0:2}|A_2, O, u_{1:2}; \boldsymbol{\Psi_{0:2}})$ giving the filtered joint distribution, Equation 5.4.4.

Our goal is to be able to compute the marginals $P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t}; \boldsymbol{\theta}_o)$ of the agent and object random variables and evidence $P(Y_{0:t}|u_{1:t}; \alpha_{0:t})$ without having to perform an expensive marginalisation over the entire space of the joint distribution as was the case for Histogram-SLAM. The next section describes how to efficiently compute the evidence and the marginals. For ease of notation we will not always show the conditioned actions $u_{1:t}$.

5.4.1 Evidence and marginals

In order to compute efficiently the marginal likelihood (also known as evidence) $P(Y_{0:t}|u_{1:t}; \alpha_{0:t})$ and the filtered marginals $P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t}; \boldsymbol{\theta}_o)$ we take advantage of the fact that only a very small area in the joint distribution space will be affected by the measurement likelihood function at each time step. Without loss of generality the likelihood function will only make a difference to dependent $A \cap O$ states in the joint distribution, states where the likelihood function is less than one. The independent states $A \ominus O$ will not be affected, where the likelihood function is equal to one. Figure 5.8 shows the relation between the measurement function $P(Y_t|A_t, O)$ and the joint distribution $P(A_t, O|Y_{0:t})$ for three different initialisations.

As illustrated and explained in Figure 5.8, the joint distribution can be decomposed in a dependent and independent term (Equation 5.4.10).

$$P(A_t, O|Y_{0:t}) = P_{\cap}(A_t, O|Y_{0:t}) + P_{\ominus}(A_t, O|Y_{0:t}) \tag{5.4.10}$$

The probability mass covered by the dependent term is located within the measurement function's tube and the independent probability mass is located outside. This formulation will lead to large computational gain as the independent term is not influenced by the measurement function: $P_{\ominus}(A_t, O, \mathbf{Y_{0:t}}) = P_{\ominus}(A_t, O, \mathbf{Y_{0:t-1}})$ and $P_{\ominus}(A_t, O|\mathbf{Y_{0:t}}) \propto P_{\ominus}(A_t, O|\mathbf{Y_{0:t-1}})$.

Evidence

The evidence of the measurement $P(Y_{0:t}|u_{1:t}; \alpha_{0:t})$ is the normalisation coefficient of the joint distribution Equation 5.4.4. It is the amount of probability mass re-normalised to the other parameters as a result of the consecutive application of the likelihood function. At time step t, the normalising factor to be added to the evidence is the difference between the probability mass located inside $A \cap O$ before and after the application of the measurement function $P(Y_t|A_t, O)$, see Equation 5.4.11-5.4.12 (see Appendix B.4 for the full derivation).

$$P(A_t, O|Y_{0:t}) = P_{\cap}(A_t, O|Y_{0:t}) + P_{\ominus}(A_t, O|Y_{0:t})$$

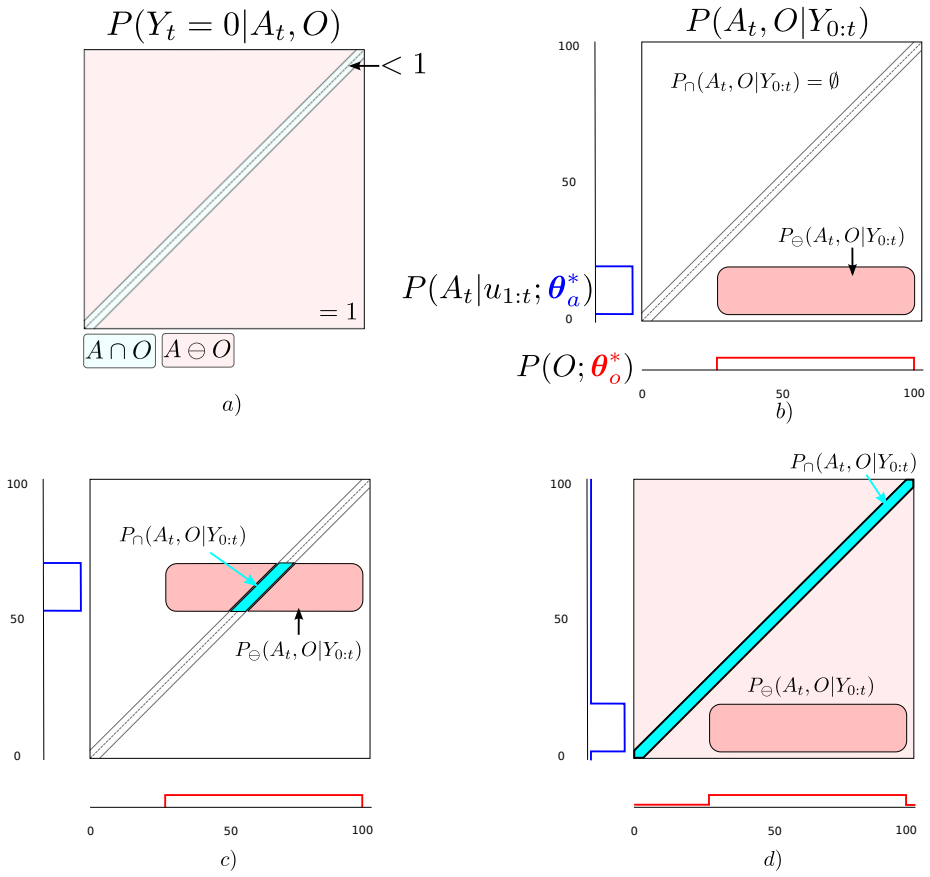


Figure 5.8: **a)** Likelihood $P(Y_t = 0|A_t, O)$, the blue area depicts the regions in which the likelihood is < 1 and the red area is where the likelihood is $= 1$. If the probability mass of the joint distribution is in the blue region, then the parameters of the random variables in these areas are dependent, $A \cap O$. Otherwise they are independent, $A \ominus O$. **b)** The agent and object marginals are not overlapping and are thus completely independent. The joint distribution, $P(A_t, O|Y_{0:t})$ the black rectangle, is not intersecting with the measurement function. As a result $P_{\cap}(A_t, O|Y_{0:t})$ is empty. **c)** The marginals overlap resulting in the measurement likelihood function intersecting with the joint distribution. The joint distribution is composed of the blue and red areas, Equation 5.4.10. The probability mass at the intersection is removed and re-normalised to other regions which is the result of applying Bayes integration. **d)** The marginals of A and O are completely overlapping, however only a small fraction of the probability mass in the joint distribution is within the measurement function's tube.

$$\alpha_t = \sum_{A_t} \sum_{O} \Big(P(Y_t|A_t, O) - 1 \Big) P_{\cap}(A_t, O, Y_{0:t-1}|u_{1:t}) \quad (5.4.11)$$

$$P(Y_{0:t}|u_{1:t}; \boldsymbol{\alpha_{0:t}}) = 1 + \underbrace{\alpha_{0:t-1} + \alpha_t}_{\alpha_{0:t}} \quad (5.4.12)$$

The advantage of Equation 5.4.11 is that the summation is only over the states which are in the dependent area $\cap$ of the joint distribution. This is generally always much smaller than the full space itself. Until an object is sensed, the likelihood will always be zero $P(Y_t|A_t, O) = 0$ and α_t will correspond to the probability mass which falls within the region of the joint distribution in which the likelihood function is zero. In Figure 5.8 b) & d), the sum of the probability mass in the blue regions is equal to α_t. The point of interest is that as we perform the filtering process we will never re-normalise the whole joint distribution, but only keep track of how much it should have been normalised. To this end the marginals $P(A_t|u_{1:t}; \boldsymbol{\theta}^*_a)$ and $P(O; \boldsymbol{\theta}^*_o)$ are never re-normalised but are used at each step to compute how much of the probability mass α_t should go to the normalisation factor $P(Y_{0:t}|u_{1:t}; \boldsymbol{\alpha_{0:t}})$. The normalisation factor in question will never be negative, as the joint distribution's sum is one and each α_t represents some of the mass removed from the joint distribution. Since we keep track of the history of applied measurement likelihood functions the same amount of probability mass is never removed twice from the joint distribution.

Marginals

There are two different sets of marginals used in the MLMF filter. The first set are the **joint marginals** of the joint distribution, Equation 5.4.4 parameterised by $\boldsymbol{\theta}^*_a$ and $\boldsymbol{\theta}^*_o$. The second set of marginals are the **filtered marginals** which are updated by evaluating the joint distribution in dependent states and are parameterised by $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_o$. At initialisation before the first action or observation is made the parameters of the filtered marginal are set equal to those of the joint distribution.

In Histogram-SLAM both the agent and object marginals are obtained, at each time step, by marginalising the joint distribution. This requires storing and summing over all the parameters of the joint distribution which is expensive. Instead the MLMF takes advantage of the sparsity of the likelihood function which results in only the dependent elements of the marginal being affected, Equation 5.4.13 (see Appendix B.5 for the full derivation of Equation 5.4.13).

$$P(O|Y_{0:\mathbf{t}}; \boldsymbol{\theta}_o) = P(O|Y_{0:\mathbf{t-1}}; \boldsymbol{\theta}_o) - \Big(P_{\cap}(O|Y_{0:t-1}) - \mathbf{P}_{\cap}(\mathbf{O}|\mathbf{Y_{0:t}}) \Big) \quad (5.4.13)$$

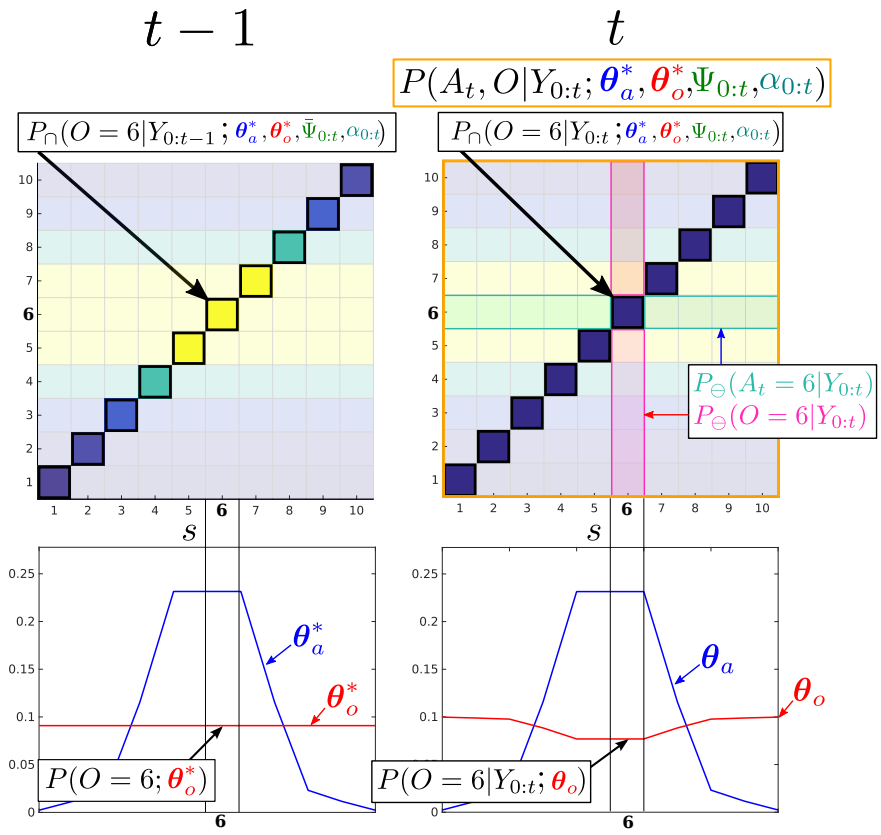


Figure 5.9: Filtered marginals. Illustration of the agent and object marginal update, Equation 5.4.13. The joint distribution parameters which are independent $A \ominus O$ are pale and the dependent areas $A \cap O$, where $P(Y_t < 1|A_t, O)$, are bright. MLMF only evaluates the joint distribution in dependent states. For each state s of the marginals $1, \ldots, 10$ the difference of the marginals inside the dependent area, before and after the measurement likelihood is applied, is evaluated and removed from the marginals $P(A_t|Y_{0:t-1}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t-1}; \boldsymbol{\theta}_o)$ leading to $P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t}; \boldsymbol{\theta}_o)$ (we did not show $u_{1:t}$ in the figure for ease of notation). *Bottom-left*: joint marginals $P(A_t|u_{1:t}; \boldsymbol{\theta}^*_a)$ and $P(O; \boldsymbol{\theta}^*_o)$ remain unchanged by measurements.

$$\mathbf{P}_\cap(\mathbf{O}|\mathbf{Y_{0:t}}; \boldsymbol{\theta}^*_a, \boldsymbol{\theta}^*_o, \boldsymbol{\Psi_{0:t}}, \boldsymbol{\alpha_{0:t}}) = \sum_{A_t} \mathbf{P}_\cap(\mathbf{A_t}, \mathbf{O}|\mathbf{Y_{0:t}}, \mathbf{u_{1:t}}; \boldsymbol{\theta}^*_a, \boldsymbol{\theta}^*_o, \boldsymbol{\Psi_{0:t}}, \boldsymbol{\alpha_{0:t}})$$

$$= \frac{\sum_{A_t} P_\cap(O; \boldsymbol{\theta}^*_o) P_\cap(A_t|u_{1:t}; \boldsymbol{\theta}^*_a) P(Y_{0:t}|A_t, O, u_{1:t}; \boldsymbol{\Psi_{0:t}})}{P(Y_{0:t}|u_{1:t}; \boldsymbol{\alpha_{0:t}})} \quad (5.4.14)$$

Equation 5.4.13 is recursive, $P(O|Y_{0:\mathbf{t}}; \boldsymbol{\theta}_o)$ is computed in terms of $P(O|Y_{0:\mathbf{t-1}}; \boldsymbol{\theta}_o)$. Figure 5.9 illustrates a measurement update of the MLMF. The illustrated marginals (*Bottom row*) are (on the *left*) the "joint marginals" $P(A_t|u_{1:t}; \boldsymbol{\theta}^*_a)$, $P(O; \boldsymbol{\theta}^*_o)$ and (on the *right*) the "filtered marginals" $P(A_t|Y_{0:t}, u_{1:t}; \boldsymbol{\theta}_a)$, $P(O|Y_{0:t}; \boldsymbol{\theta}_o)$.

The shape of the joint marginals remain unchanged by measurements during the filtering process, they are the parameters of the joint distribution used to update the filtered marginals. Table 5.1 summarises the functions and parameters

of the MLMF for two random variables, an agent and object.

functions		parameters	description
$P(A_t \mid Y_{0:t}, u_{1:t})$	:	$\boldsymbol{\theta}_a$	filtered marginals
$P(O \mid Y_{0:t})$	:	$\boldsymbol{\theta}_o$	
$P(A_t \mid u_{1:t})$	:	$\boldsymbol{\theta}_a^*$	joint marginals
$P(O)$	:	$\boldsymbol{\theta}_o^*$	
$P(Y_{0:t} \mid u_{1:t})$	:	$\alpha_{0:t} \in \mathbb{R}$	evidence
$P(Y_{0:t} \mid A_t, O, u_{1:t})$	:	$\Psi_{0:t} = \{(Y_i, l_i)\}_{i=0:t}$	likelihood history

Table 5.1: MLMF functions with associated parameters. The marginal parameters are the discretisation of the state space $\boldsymbol{\theta} \in \mathbb{R}^N$, $\boldsymbol{\theta}^{(s)}$ correspond to the probability being in state s.

We evaluated the MLMF with Histogram-SLAM in the case of the 1D filtering scenario illustrated in Figure 5.6 on page 132 and we found them to be identical. Having respected the formulation of Bayes rule, we assert that the MLMF filtering steps (see Algorithm 3, Appendix B.6 for a more detailed application of motion-measurement update steps) are Bayesian Optimal Filter[1]. Next we evaluate both space and time complexity of the MLMF filter.

5.4.2 Space & time complexity

For discussion purposes we consider the case of three beliefs, namely that of the agent and two other objects $O^{(1)}$ and $O^{(2)}$, we subsequently generalise. As stated previously M stands for the number of filtered random variables including the agent and N is the number of discrete states in the world. In the following section, we compare the space and time complexity of MLMF-SLAM with Histogram-SLAM.

Space complexity

Figure 5.10 *Left* illustrates the volume occupied by the joint distribution for a space with N states. Histogram-SLAM would require N^3 parameters for the joint distribution $P(A, O^{(1)}, O^{(2)}; \boldsymbol{\theta})$ and $3\,N$ parameters to store the marginals. In general for M random variables $N^M + M\,N$ parameters are necessary, give a space complexity of $\mathcal{O}(N^M)$.

For MLMF-SLAM, each random variable requires two sets of parameters, $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^*$ (see Table 5.1). Given M random variables, the initial number of parameters is $M(2N)$. At every time step the likelihood memory function increments by one measurement and offset, $(Y_t, l = 0)$ (Algorithm 2). Given a state space of size N, there can be no more than N different measurement functions (one for each state). In the worst case scenario the number of memory likelihood

[1] An optimal Bayesian solution is an exact solution to the recursive problem of calculating the exact posterior density Arulampalam et al. (2002)

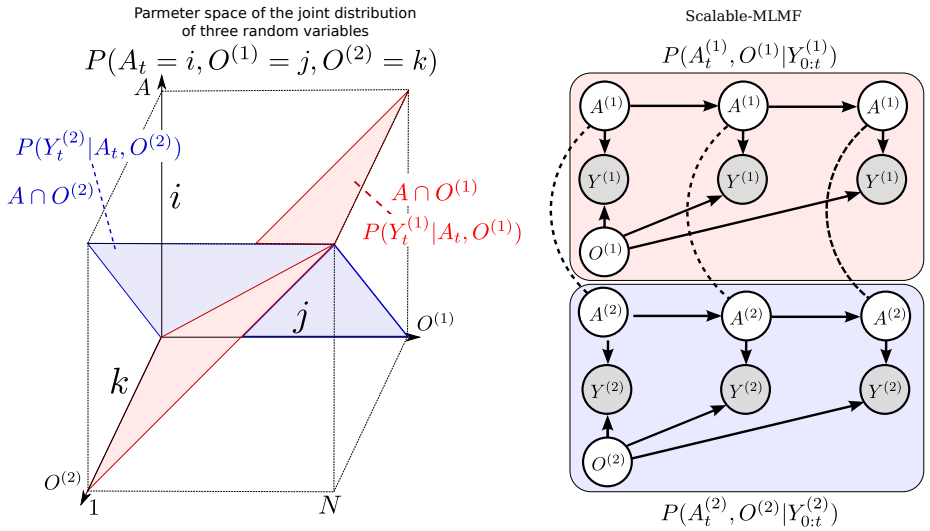


Figure 5.10: *Left:* Joint distribution $P(A, O^{(1)}, O^{(2)})$ of the agent and two objects. Each measurement likelihood function, $P(Y|A, O^{(1)})$, $P(Y|A, O^{(2)})$ corresponds to a hyperplane in the joint distribution The state space is discretised to N bins giving the total number of parameters in the joint distribution of N^3. *Right:* **Scalable-MLMF** Each agent-object joint distribution pair is modelled independently. For clarity we have left out the action random variable u which is linked to every agent node. Two joint distributions $P(A^{(1)}, O^{(1)}|Y_{0:t}^{(1)})$ and $P(A^{(2)}, O^{(2)}|Y_{0:t}^{(2)})$ parametrise the graphical model. The dashed undirected lines represent a wanted dependency, if present $O^{(1)}$ and $O^{(2)}$ are to be dependent through A. In the standard setting there will be no exchange of information between the individual joint distributions. However we demonstrate later on how we perform a one time transfer of information when one of the objects is sensed.

function parameters $\Psi_{0:t}$, Equation 5.4.7, will be N. The total number of parameters is $M(2N) + N$ which gives a final worst case space complexity which is linear in the number of random variables, $\mathcal{O}(N\,M)$.

Time complexity

For Histogram-SLAM, the computational cost is equivalent to that of the space complexity, $\mathcal{O}(N^M)$, since every state in the joint distribution has to be summed to obtain all the marginals.

For MLMF-SLAM, every state in the joint distribution's state space which has been changed by the likelihood function has to be summed, see Figure 5.9 on page 140. As a result the computational complexity is directly related to the number of dependent states $|A \cap O|$. In Figure 5.9, this corresponds to states where $i = j$ and there are N out of a total N^2 states for that joint distribution. Figure 5.10 (*Left*) illustrates a joint distribution with N^3 states. The dependent states $|A \cap O^{(1)} \cap O^{(2)}|$ are those which are within the blue and red planes (where the likelihood evaluates to zero) and comprise N^2 states each, giving a total of $2\,N^2 - N$ dependent states (negative is to remove the states we count twice at the intersection of the blue and red plane).

The likelihood term $P(Y_t|A_t, O^{(1)})$ evaluates states to zero which satisfy $(i = j, \forall k)$, as the measurement of object $O^{(1)}$ is independent of object $O^{(2)}$. With 3 objects, the joint distribution would be $P(A_t = i, O^{(1)} = j, O^{(2)} = k, O^{(l)} = l)$ then the likelihood $P(Y_t|A_t, O^{(1)})$ evaluated to zero for $(i = j, \forall k, \forall l)$ which would mean N^3 dependent states. In general, for M random variables the computational cost is $(M-1)N^{M-1}$ which gives $\mathcal{O}(N^{M-1})$ as opposed to the Histogram-SLAM's $\mathcal{O}(N^M)$. The computation complexity in this setup is still exponential but to the order $M-1$ as opposed to M which nevertheless quickly limits the scalability as more objects are added.

Computing the value of a dependent state (i, j, k) in the joint distribution required evaluating Equation 5.4.4 which contains a product of N likelihood functions, in the worst case scenario. However the likelihood functions are not overlapping and binary. As a result the complete product does not have to be evaluated since only one likelihood function will effect the state (i, j, k). Thus evaluating Equation 5.4.4 yields a cost of $\mathcal{O}(1)$ and not $\mathcal{O}(N)$.

5.4.3 Scalable extension to multiple objects

To make the MLMF filter scalable we introduce an **independence assumption** between the objects and model the joint distribution (Equation 5.4.15) as a product of agent-object joint distributions:

$$P(A_t, O^{(1)}, \cdots, O^{(M-1)}|Y_{0:t}, u_{1:t}) = \prod_{i=1}^{M-1} P(A_t^{(i)}, O^{(i)}|Y_{0:t}^{(i)}, u_{1:t}) \tag{5.4.15}$$

The measurement variable Y_t, is the vector of all agent-object measurements, $Y_t = \left[Y_t^{(1)}, \ldots, Y_t^{(M-1)}\right]^{\mathrm{T}}$. Each agent-object joint distribution has its own parametrisation of the agent's marginal, $A_t^{(1)}, \ldots, A_t^{(M-1)}$ which combine to give the overall marginal of the agent A_t. The computation of each object marginal $P(O^{(i)}|Y_{0:t}^{(i)})$ is independent of the other objects. This is evident from the marginalisation see Equation 5.4.16-5.4.17.

$$P(O^{(i)}|Y_{0:t}^{(i)}, u_{1:t}) = \sum_{A_t^{(i)}} P(A_t^{(i)}, O^{(i)}|Y_{0:t}^{(i)}, u_{1:t}) \tag{5.4.16}$$

$$P(A_t|Y_{0:t}, u_{1:t}) = \prod_{i=1}^{M-1} P(A_t^{(i)}|Y_{0:t}^{(i)}, u_{1:t}) \tag{5.4.17}$$

The independence assumption will create an unwanted effect with respect to agent's marginal $P(A_t|Y_{0:t}, u_{1:t})$. At initialisation the agent marginals should be equal, $P(A_0|Y_0) = P(A_0^{(i)}|Y_0^{(i)})\forall_i$, however this is not the case because of

	space	time
Histogram	$\mathcal{O}(N^M)$	$\mathcal{O}(N^M)$
MLMF	$\mathcal{O}(M\,N)$	$\mathcal{O}(N^{(M-1)})$
scalable-MLMF	$\mathcal{O}(M\,N)$	$\mathcal{O}(M\,N)$

Table 5.2: Time and space complexity summary For both MLMF and scalabe-MLMF the worst case scenario is reported for the space complexity.

Equation 5.4.17. To overcome this we define the marginal, $P(A_t|Y_{0:t}, u_{1:t})$, of the agent as being the average of all the individual pairs $P(A^{(i)}|Y^{(i)}_{0:t}, u_{1:t})$.

$$P(A_t|Y_{0:t}, u_{1:t}) := \frac{1}{M-1}\sum_{i=1}^{M-1} P(A^{(i)}_t|Y^{(i)}_{0:t}, u_{1:t}) \tag{5.4.18}$$

Figure 5.10 (*Right*), depicts the graphical model of the scalable formulation. As each joint distribution pair has its own parametrisation of the agent's marginal and these do not subsequently get updated by one another, the information gained by one joint distribution pair is **not transferred**. A solution is to transfer information between the marginals $A^{(i)}$ at specific intervals namely when one of the objects is sensed by the agent.

The exchange of information of one joint distribution to another is achieved through the agent's marginals $A^{(i)}$ according to Algorithm 4, Appendix B.7. The measurement update is the same as previously described in Algorithm 3 in the case of no positive measurements of the objects. If the agent senses an object, all of the agent marginals of the remaining joint distributions are set to the marginal of the joint distribution pair belonging to the positive measurement $Y^{(i)}_t$.

Figure 5.11, depicts the process of information exchange between object $O^{(1)}$ and $O^{(2)}$ in the event that the agent senses $O^{(2)}$. Prior to the positive detection, both marginals $P(A^{(1)}_t|Y^{(1)}_{0:t-1}, u_{1:t})$ and $P(A^{(2)}_t|Y^{(2)}_{0:t-1}, u_{1:t})$ occupy the same region and are identical. When the agent senses $O^{(2)}$ the line defined by the measurement likelihood function $P(Y^{(2)}_t|A^{(2)}_t, O^{(2)})$ becomes a hard constraint implying that both the agent and $O^{(2)}$ have to satisfy this constraint. Figure 5.12 shows marginals at initialisation, prior contact between the agent and object and the after the measurement (post contact) has been integrated into the marginals (resulting from the joint distributions in Figure 5.11).

The result of introducing a dependency between the objects through the agent's marginals in the event of a sensing and treating them independently gives the same solution as the histogram filter in this particular case. However as each individual marginal $A^{(i)}_t$ diverges from the other marginals, the filtered solution will diverge from the histogram's solution. We assume however that the objects are weakly dependent and sharing information during positive sensing events is sufficient. In section 5.5.2 we will evaluate the independence assumption with respect to the histogram filter.

Table 5.2 summarises the time and space complexity for the three filters.

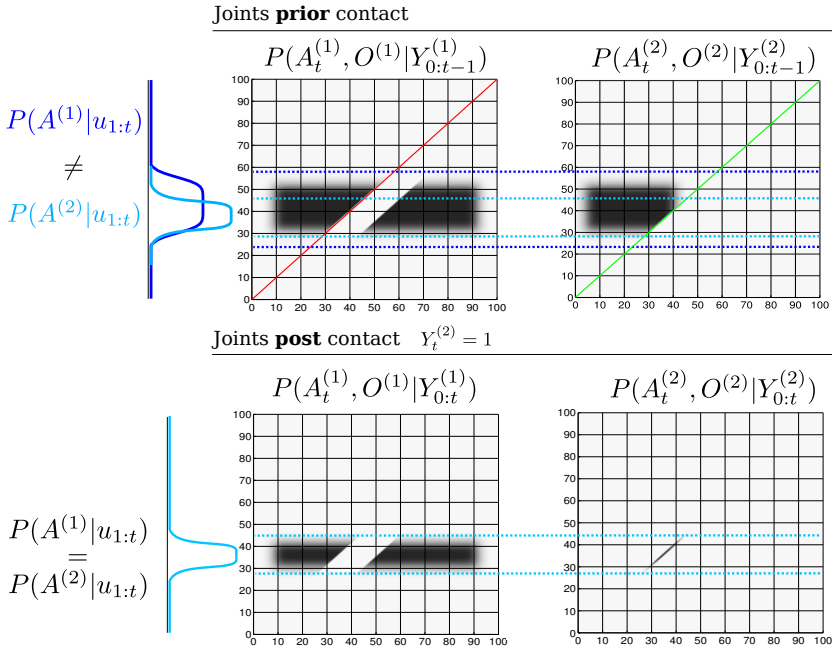


Figure 5.11: Transfer of information (joint distributions) *Top:* Joint distributions of $P(A_t^{(1)}, O^{(1)}|Y^{(1)})$ and $P(A_t^{(2)}, O^{(2)}|Y^{(2)})$ prior sensing, $Y_t^{(2)} = 1$, see Figure 5.12 (*Top right*) for the corresponding marginals. The red and green lines across the joint distributions correspond to the region in which the likelihood functions $P(Y_t^{(1)}|A_t^{(1)}, O^{(1)})$ and $P(Y_t^{(2)}|A_t^{(2)}, O^{(2)})$ will change the joint distributions. The dotted blue lines are to ease the comparison ofthe joint distributions prior and post sensing. *Bottom right:* After the agent has sensed $O^{(2)}$, all the probability mass which was not overlapping the green line becomes an infeasible solution to the agent and object locations. At this point the marginals $P(A_t^{(1)}|u_{1:t}) \neq P(A_t^{(2)}|u_{1:t})$ are no longer equal (see the blue marginals *Top*). *Bottom left:* The constraint imposed by the likelihood function of the second object (green line) is transferred to the joint distribution of the first object according to Algorithm 4. This results in a change in the joint distribution $P(A_t^{(1)}, O^{(1)}|Y^{(1)})$, which satisfies the constraints imposed by the agent's marginal from the joint distribution $P(A_t^{(2)}, O^{(2)}|Y^{(2)})$.

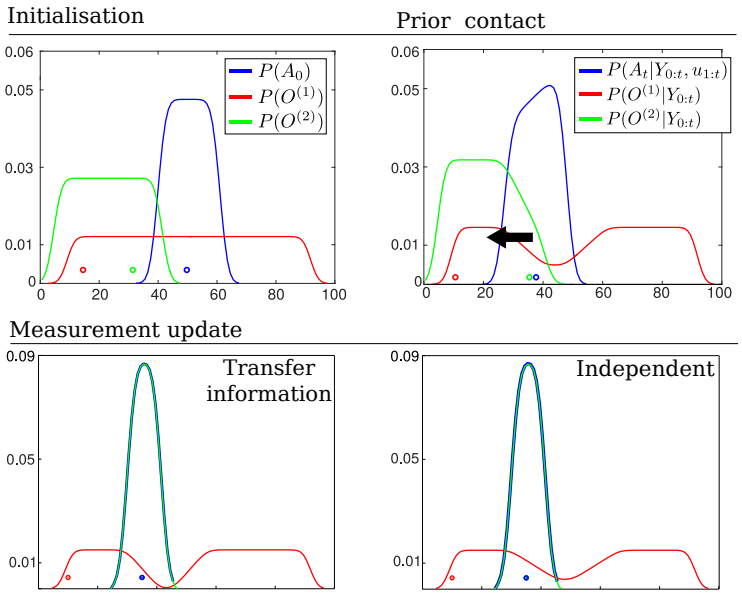


Figure 5.12: Transfer of information (marginals) *Top left:* Initial beliefs of the agent and object's location. The agent moves to the left until it senses object $O^{(2)}$. *Top right:* Marginals prior the agent entering in contact with the green object, see Figure 5.11 (*Top*) for an illustrate of the joint distributions. *Bottom left:* resulting marginals after setting the agent marginals of each joint distribution equal $A_t^{(1)} = A_t^{(2)}$ according to Algorithm 4. The object marginal $P(O^{(2)}|Y_{0:t})$ is recomputed. *Bottom Right:* resulting marginals in which the objects have no influence on one another. Note that a transfer of information has caused a change in the marginal $O^{(1)}$.

5.5 Evaluation

We conduct three different types of evaluation to quantify the scalability and correctness of the scalable-MLMF filter. The first experiment tests the scalability of our filter in terms of processing time taken per motion-measurement update cycle. The second experiment evaluates the independence assumption made in the scalable-MLMF filter between the objects. The third and final experiment determines the effect of the memory size on a search policy to locate all the objects in the *Table* world.

5.5.1 Evaluation of time complexity

We measured the time taken by the motion-measurement update loop, as a function of the number beliefs and number of states per belief. We started with a 100 states per belief and gradually increased it to 10'000'000 over 50 steps. Each of the 50 steps treated 2 to 25 objects. Figure 5.13 *left* illustrates the computational cost as a function of number of states and objects. For each state-object pair 100 motion-measurement updates were performed. Most of the trials returned time updates below 1 Hz. Figure 5.13 *right* shows the computational cost as a function of the number of states plotted for 6 different

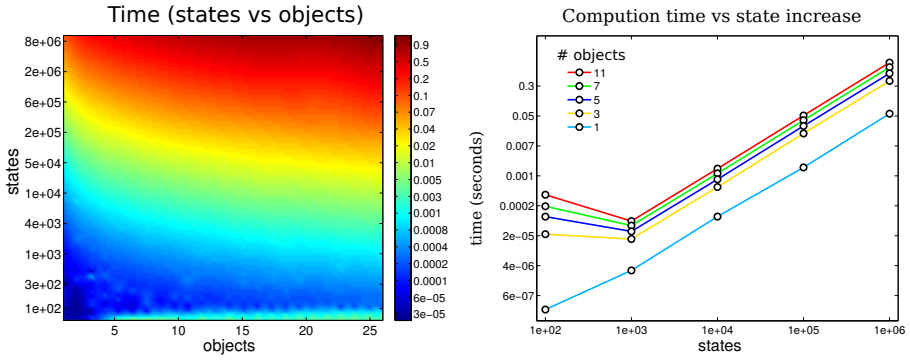


Figure 5.13: Time complexity: *left:* mean time taken for a loop update (motion and measurement) as a function of the number of states in a marginal and the number of objects present. *right:* time taken for a loop update with respect to the number of states in the marginal. The colour coded lines are associated with the number of objects present. The computational cost is plotted on a log scale. As the number of states increases exponentially the computational cost matches it.

filter runs with a different number of objects. As the number of states increases exponentially so does the computational cost. Note the cost increases at the same rate as the number of states meaning that the computational complexity is linear with respect to the number of states. This result is in agreement with the asymptotic time complexity.

5.5.2 Evaluation of the independence assumption

In Section 5.4.3 we made the assumption (for scalability reasons) that the objects' beliefs are independent of one another. This assumption is validated by comparing the MLMF filter on three random variables, an agent and two objects, with the ground truth which we obtain from the standard histogram filter. For each of the three beliefs (the agent and two objects), 100 different marginals were generated and the true locations (actual position of the agent and objects) were sampled. Figure 5.14 *Top-left* illustrates one instance of the initialisation of the agent and object marginals with their associated sampled true position. The agent carries out a sweep of the state space for each of the marginals and the policy is saved and run with the scalable-MLMF filter. In the first experiment we assumed that the objects are completely independent and that there was no transfer of information between the pair-wise joint distributions. In the second and third experiments there is an exchange of information as described in Algorithm 4. Here we compare the effect of using the product of the agent's marginals, Equation 5.4.17, with the average of the marginals, Equation 5.4.18. We expect the average of the the agent's marginal to yield a result closer to the ground truth as the marginal of the agent $P(A_t|Y_{0:t}, u_{1:t})$ at initialisation is the same as the ground truth (the Histogram-SLAM's). As for the marginal of the

objects $P(O^{(i)}|Y_{0:t})$ we expect the difference between them to be independent of whether the product or average of the agent's marginal is used. This results from Algorithm 4. When an object i is sensed all the corresponding agent marginals $P(A^{(j)}|u_{1:t})$ are set equal to $P(A^{(i)}|u_{1:t})$ and not to $P(A_t|Y_{0:t}, u_{1:t})$. This is a design decision of our information transfer heuristic. There are many other possibilities but this is one of the simplest. For each of the 100 sweeps the ground truth is compared with the scalabe-MLMF using the Hellinger distance (Equation 5.5.1)

$$H(P,Q) = \frac{1}{\sqrt{2}} \|\sqrt{P} - \sqrt{Q}\|_2 \tag{5.5.1}$$

which is a metric which measures the distance between two probability distributions. Its value lies strictly between 0 (the two distributions are identical) and 1 (no overlap between them). Figure 5.14 shows the kernel density distribution of the Hellinger distances taken at each time step for all 100 sweeps. In the *Top-left* of the figure, for the case when no transfer of information is applied, all the marginals are far from the ground truth. This results from the introduction of the independence assumption, necessary to scale the MLMF. Figure 5.14 *Bottom* shows the results for difference between the product and average of the agents marginals. As expected there is no difference between the objects' marginals when considering both methods (product and average) with respect to the ground truth. The predominant difference occurs in the agent's marginal $P(A_t|Y_{0:t}, u_{1:t})$. This is also expected and prompted the introduction of the average method instead of the product.

The scalable-MLMF information exchange heuristic will not lead to any of the objects marginals probability mass being falsely removed during the information transfer, which is close to a winner-take-all approach in terms of beliefs. When object i is sensed its associated agent marginal is set to all other agent-object joint pairs, which results in the information accumulated in the jth agent marginals being replaced by the ith.

5.5.3 Evaluation of memory

The memory measurement likelihood function $P(Y_{0:t}|A_t, O, u_{1:t}; \Psi_{0:t})$ is parameterised by the history of all the measurement likelihood functions which have been applied on the joint distribution since initialisation. As detailed previously there can be no more than $|\Psi_{0:t}| \leq N$ different measurement likelihood functions added to memory. In the case of a very large state space this might be cumbersome. We investigate how restricting the memory size, the number of parameters $|\Psi_{0:t}|$, can impact on the decision process in an Active-SLAM setting. Given our set up a breadth-first search in the action space is chosen with a one time step horizon, making it a greedy algorithm. The objective function utilised is the information gain of the beliefs after applying an action, Equation 5.5.2.

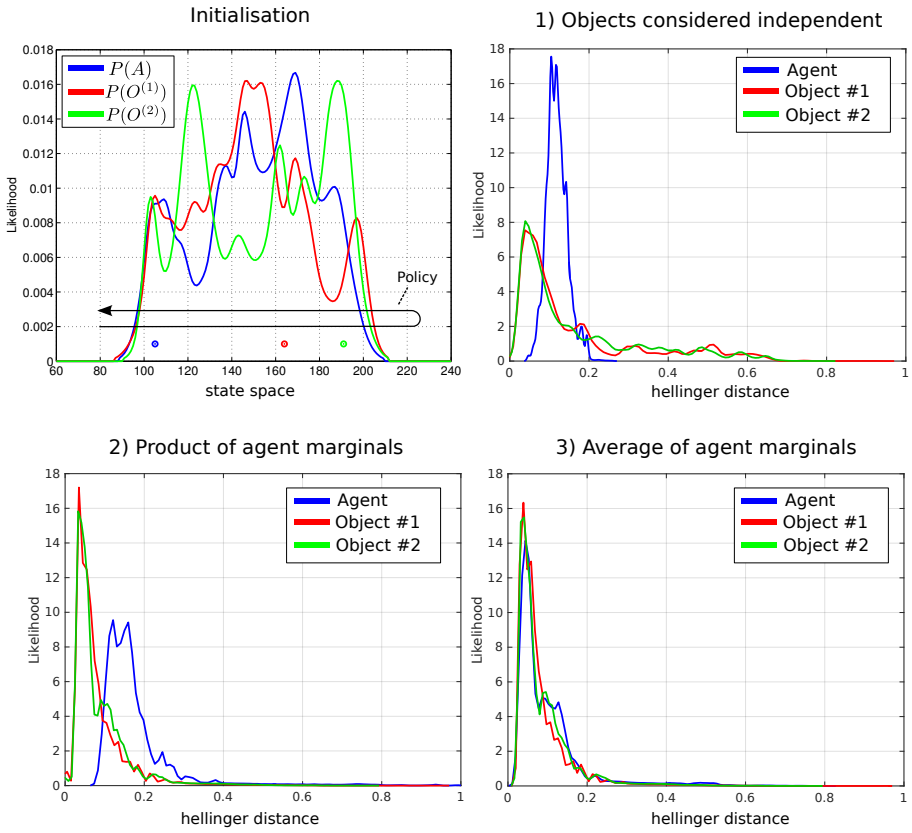


Figure 5.14: Comparison of scalable-MLMF and the histogram filter A deterministic sweep policy was carried out for 100 different initialisations of the agent and object beliefs. *Top left:* One particular Initialisation of the agent and object random variables. The true position of the agent and objects were sampled at random. The black arrow indicates the general policy which was followed for each of the 100 sweeps. These were performed for **1)** scalable-MLMF with objects considered to be independent at all times (Algorithm 4). **2)** Agent marginal $P(A_t|Y_{0:t}, u_{1:t})$ is the product of marginals $P(A_t^{(i)}|Y_{0:t}^{(i)}, u_{1:t})$, Equation 5.4.17. **3)** marginal $P(A_t|Y_{0:t}, u_{1:t})$ is taken to be the average of all marginals $P(A_t^{(i)}|Y_{0:t}^{(i)}, u_{1:t})$, Equation 5.4.18. For each of these three experiment we report the kernel density estimation over the Hellinger distances taken at every time step between ground truth (from histogram filter) and scalable-MLMF.

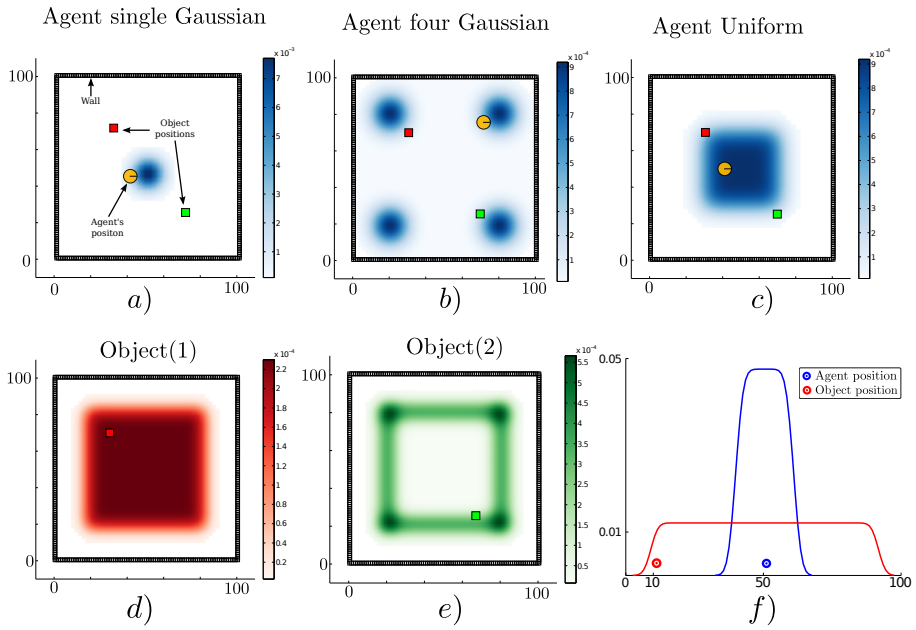


Figure 5.15: Agent's prior beliefs. Two types of environment, the first is a 2D world where the agent lives in a square surrounded by a wall whilst the second is a 1D world. In the 2D figures the agent is illustrated by a circle with a bar to indicate its heading. The true location of the objects are represented by colour coded squares. *Top row* three different initialisations of the agent's location. *Bottom row* d) the agent's prior beliefs with respect to the location of the first object and e) belief of the second object's location. *bottom row* f) 1D world with one object.

$$u_t^* = \arg\max_{u_t} H\{P(A_{t-1}, O|Y_{0:t-1}, u_{1:t-1})\} - \mathbb{E}_{Y_t}\left[H\{P(A_t, O|Y_{0:t}, u_{1:t})\}\right] \tag{5.5.2}$$

For each action the filter is run forward in time and all future measurements since we cannot know ahead of time the actual measurement. The information gain is the difference between the current entropy (defined by $H\{\cdot\}$) and the future entropy after the simulated motion and measurement update. The action u_t^* with the highest information gain is subsequently selected. This is repeated at each time step. Figure 5.15 illustrates the environment setup for a 1D and 2D case. The agent's task is to find the objects in the environment.

For the 2D search we consider three different initialisations (single-Gaussian, four-Gaussian, Uniform) for the agent's belief where there are two objects to be found. Ten searches are carried out for each of the three initialisations of the agent's beliefs. The agent's true location, for each search, is sampled from its initial belief, and the objects' locations (red and green squares in Figure 5.15) are kept fixed throughout all searches. Each search is repeated for 18 different memory sizes ranging from 1 to N (the number of states). For the 1D search case one object is considered since adding more objects makes the search easier and the interest lies in the memory effects of the search and not the search

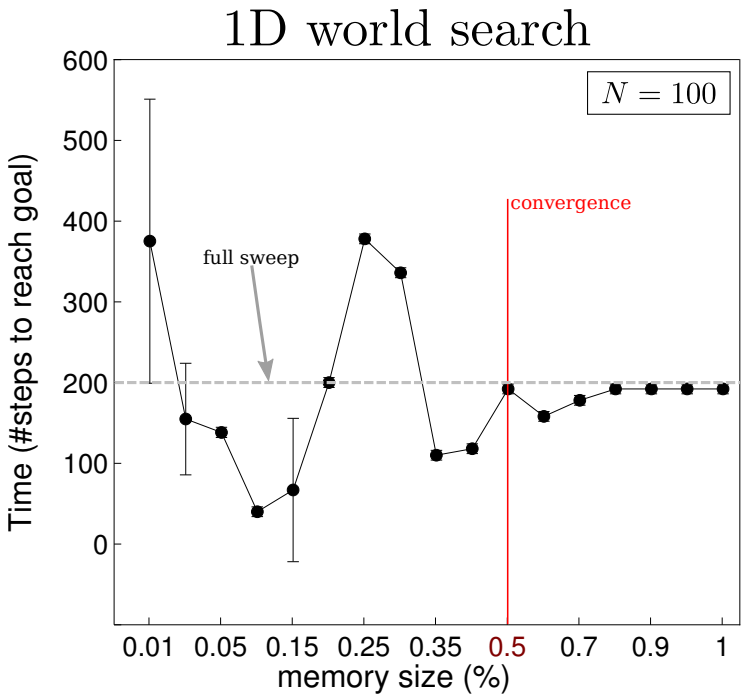


Figure 5.16: Memory size vs time to find object in 1D Results of the effect of the memory size on the decision process for the 1D search illustrated in Figure 5.15 *f)*. The memory size is reported as the percentage of total number of states present in the marginal space. At 100% the size of the memory is equal to that of the state space, $N = 100$ in this case. A total sweep of the entire state space would result in a total of 200 steps, the dotted grey line in the above figure. When no restrictions are placed on the memory size the policy following the greedy approach takes around 180 steps. This result converges when the number of parameters $|\Psi_{0:t}|$ of the memory likelihood function is greater than 50% of the original state space.

itself. In Figures 5.16-5.17 we report on the time taken to find all objects with respect to a given memory size which is shown as the percentage of the total number of states. In the 1D search case the time variability taken to find the object converges when the memory size is at 60% of the original state space. As for the 2D search with 2 beliefs (agent & 1 object) the convergence depends on the agent's initial belief. For the 1-Gaussian (green line) all searches take approximately the same amount of time after a memory size of 9%. As for the remaining two initialisations convergence is achieved at 48%. The same holds true for the case of 3 beliefs (agent & 2 objects).

In the 2D searches, the memory size has a less impact on the time taken to find the objects than in the 1D (which is a special search case). Only when the memory size is less than 6% is there a significant change. We conclude that at least in the case of the greedy one step-look ahead planner which is frequently used in the literature, the size of the memory seems not to be a limiting factor in terms of the time taken to accomplish the search.

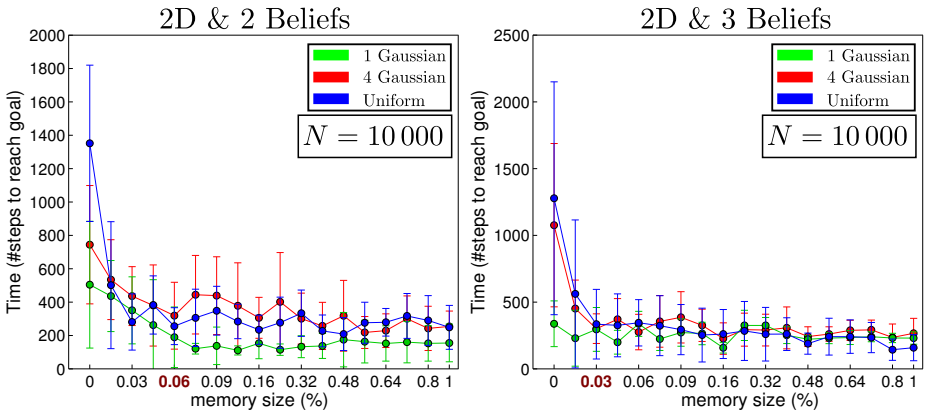


Figure 5.17: Memory size vs time to find objects in 2D. The initial beliefs correspond to those of Figure 5.15, a) for Gaussian (green line), b) 4 Gaussians (red line) and c) Uniform (blue line), both objects are initialised according to d) and e).

5.6 Conclusion

This work addresses the Active-SLAM filtering problem for scenarios in which sensory information relating to the map is very limited. Current SLAM algorithms filter the errors originating from sensory measurements and not prior uncertainty. By making the assumption that the joint distribution of all the random variables is a multivariate Gaussian, inference is tractable. Since the origin of the uncertainty does not originate from the measurement noise, no assumption can be made about the structure of the joint distribution. In this case a suitable filter would be the histogram which makes no assumption about the shape or form taken by the joint distribution. However, the space and time complexity are exponential with respect to the number random variables and this is a major limiting factor for scalability.

The main contribution of this work is a formulation of a histogram Bayesian state space estimator in which the computational complexity is both linear in time and space. A different approach to other SLAM formulations as been taken in the sense that the joint distribution is not explicitly parameterised avoiding the exponential increase in parameter space which would otherwise have been the case. The MLMF parameters consist of the marginals and the history of measurement functions which have been applied. By solely evaluating the joint distribution at the states which are affected by the current measurement function whilst taking into account the memory, the MLMF filter obtains the same filtered marginals as the histogram filter. Further, the worst case space complexity is linear rather than exponential and the time complexity remains exponential but increases at lower rate than in the histogram filter. In striving to make the filter scalable we make the assumption that the objects are independent. An individual MLMF is used for each agent-object pair. We evaluate the difference between the scalable-MLMF with a ground truth provided by the

histogram filter for 100 different searches with respect to the Hellinger distance. We conclude that the divergence is relatively small and thus the scalable-MLMF filter provides a good approximation to the true filtered marginals. We evaluate the time taken to perform a motion-update loop for different discretisations of the state space (100 to 10'000'000 states) and number of objects (2 to 25). In most of the cases we achieve an update cycle rate below 1Hz. We evaluate how the increase of the number of states affects the computational cost and find the relationship to be linear and thus in agreement with our analysis of the asymptotic growth rate. We analyse the effect of the memory size (the remembered number of measurement likelihood functions) on the decision theoretic process of reducing the uncertainty of the map and agent during a search task. We conclude that in the 2D case the memory size has much less effect than in the 1D case and that it is not necessary to remember every single measurement function.

This implies that the MLMF and scalable-MLMF that we have are a computationally tractable means of performing SLAM in a case scenario in which mostly negative information is present and the joint distribution cannot be assumed to have any specific structure. Furthermore, the filter can be used at a higher cognitive level than the processing of raw sensory information as is often the case in Active-SLAM. MLMF would be well suited for reasoning tasks where the robot's field of view is limited.

An interesting future extension could be to make the original MLMF filter scalable without introducing assumptions. One possibility could be to consider Monte Carlo integration methods for inference. These can scale well to high dimensional spaces whilst still providing reliable estimates. A second possibility could be to investigate the use of Gaussian Mixtures as a form of parameterisation of the marginals to blend our filter with EKF-SLAM. This would allow the parameters to grow quadratically with respect to the dimension of the marginal space as opposed to exponentially as is the case with the histogram and MLMF filters.

Chapter 6

Conclusion and summary

This Chapter highlights the contributions, limitations and personal insights of the author in this thesis and proposes possible directions for future research.

6.1 Main Contributions

This thesis specifically addressed decision making by agents under uncertainty. In the field of Robotics and Artificial Intelligence (AI), considering uncertainty in search policies is not straightforward. This is due to the complexity involved in solving Partially Observable Markov Decision Processes (POMDP), commonly used to describe uncertainty in tasks, which become quickly infeasible for even the simplest problems. Although there has been progress in the development of POMDP solvers with demonstrated applications to robotics, these are predominantly verified in simulation where the action space is discretised.

This thesis offers an approach to solving POMDP problems with continuous action space and high levels of uncertainty which are non-Gaussian. To tackle the sheer complexity yielded by a continuous space we used human demonstrations to provide a set of behaviours which are encoded in a generative model used as a policy. This thesis provides three major contributions to current research which we summarise below.

First, we demonstrated a Programming by Demonstration POMDP framework (PbD-POMDP) to learn a mixture of search strategies from human teachers. This method allowed to extract multiple search strategies from a group of human teachers. The behaviour in question was predominantly risk-averse when compared to a myopic greedy policy. Even when the human subjects were localised they would keep close to salient features whilst "en route" to the goal. The Gaussian Mixture Model (GMM) policy of the belief-state action space allowed to encode the main behaviours in a continuous POMDP policy avoiding the discretisation of actions, states and time. The PbD-POMDP was able to outperform both myopic and coastal navigation algorithms thus demonstrating the usefulness of the human teachers' foresight and how to leverage it in a POMDP policy avoiding the costly uninformed exploration methods traditionally used to solve POMDP problems.

Second, we proposed a Reinforcement Learning (RL) extension in which

the task is explicitly described by a binary reward function. We demonstrated that by combining the reward function and the explorative-exploitative data, provided by human teachers, in a fitted actor-critic RL framework a significantly improved policy can be obtained after one iteration of policy evaluation and improvement. This mitigates the need of time consuming rollouts which is the traditional bottleneck in RL. We utilised a modified version of the EM algorithm (Q-EM) which weights individual demonstration data points by either the Q or advantage function. This approach allows to take into account the quality of the demonstrations, which was not the case in the PbD-POMDP approach. We demonstrated that when solely the worst two teachers' data was used, our RL method, RL-PbD-POMDP, outperformed our previous PbD-POMDP approach at a peg-in-hole task.

Third, we developed a Bayesian State Space Estimator (BSSE), which we name Measurement Likelihood Memory Filter (MLMF), to solve the Simultaneous Localisation and Mapping (SLAM) problem where only haptic information is available. This filter does not make any assumptions on the structure of the marginal distributions, they do not need to be Gaussian (EKF-SLAM). The MLMF is parameterised by the function parameters of the likelihood functions as oppose to their values as it is the case for Histogram-SLAM. We demonstrated that the MLMF is able to overcome the inherent exponential space and time complexity present in its Histogram counterpart. To the best of our knowledge this is the first work which considers "negative information" (the absence of positive observation of landmarks) in a SLAM setting.

6.2 Limitations and Future Work

We summarise the current perceived limitations of our work and discuss different approaches to resolving them.

Gaussian Mixture Controller

Throughout this dissertation, we used Gaussian Mixture Models (GMM) to encode a vector field policy which is a function of the current belief state. For the robot to be able to localise itself the policy has to be guided towards salient areas of the state space, such as edges and corners. However, as the GMM learning is non-deterministic and data driven, there are no guarantees that behaviour such as remaining in contact with edges in order to get haptic feedback will be encoded and reproduced by the GMM policy.

It would require only one poorly demonstrated trajectory which failed to establish a contact with an edge, for the vector field to be shifted resulting in a policy which fails to establish contact with the environment. Figure 6.1 illustrates an example of this drawback.

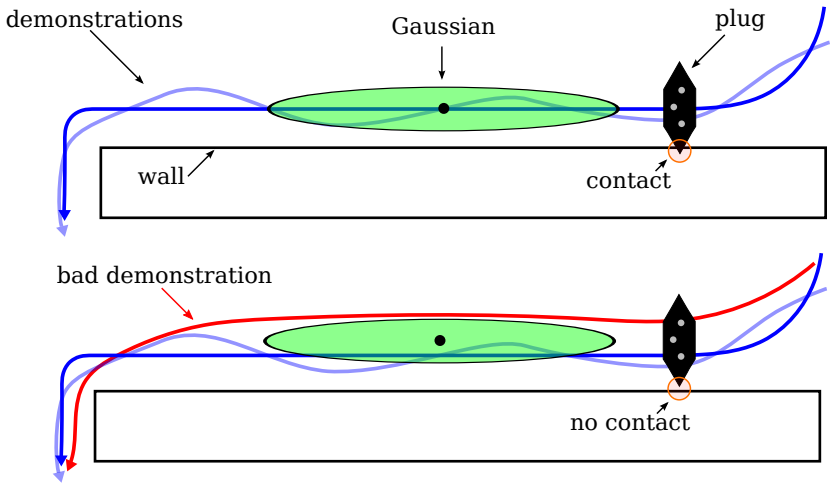


Figure 6.1: GMM policy drawback. *Top:* Two demonstrations, blue trajectories, follow a path which when fitted with a GMM (one Gaussian component is drawn in green) results in a vector which keeps the plug in contact with the edge of the wall. *Bottom:* A third demonstration, red line, did not result in a contact with the edge of the wall. The Gaussian of the GMM is shifted to the mean point of the data. The GMM policy no longer keeps the plug in contact with the edge.

A solution would be to design a cost function which gives more importance to data points which are close to salient features, such as edges and corners. As a result, the components of the problematic red trajectory in Figure 6.1 are excluded during the learning. This can be integrated into the Actor-Critic Reinforcement Learning (RL) framework used in Chapter 4, for the peg-in-hole task.

RL is an off-line approach (in the sense that many rollouts are needed to achieve the desired behaviour) which can be used to select behaviour which will result in a policy remaining in contact with the environment. However, if the geometry of the environment was to change by less than a centimetre the same situation would occur (failure to remain in contact) and GMM policy parameters would have to be relearned through either new demonstrations or via autonomous exploration.

In the peg-in-hole task described in Chapter 4, in order to enforce a constant contact with the environment, a hybrid force/position controller was used which disregarded the velocity component orthonormal to the wall. The remaining two velocity components where obtained from the GMM policy and modulated by a heuristic function to surmount the edges of the plug.

The main problem arises however during the execution of the GMM policy when no sensory feedback constraints are resolved. Belief space planners (reviewed in Chapter 2) are approaches which take into consideration variations in the environment and which can produce trajectories which actively seek sensory feedback. These planners solve an objective function online, with contact constraints Vien and Toussaint (2015b). Such online belief space planners are computationally expensive and require simulation of dynamics and sensory feed-

back in order to find an appropriate set of actions. A dynamic system approach, such as the GMM policy, learns the sensory-control mapping directly making it computationally cheap at runtime.

Future research could consider local adaptation of the dynamical system in order to try and seek out specific sensory feedback. This could be achieved for instance by combining planning with GMM policies or learning local dynamical systems which seek out sensory feedback. The difficulty lies in combining their joint actions.

Belief compression

Throughout this research, the belief is represented by a probability density function and the GMM policy is learned from a dataset with a fixed number of dimensions, which is seven in our case (3 for velocity, 3 for the most likely state and 1 for entropy). The compression of the belief to a belief state vector results in loss of information which weakens the Markov assumption. There exist however other compression methods, such as E-PCA Roy and Gordon (2003). This is a non-linear dimensionality reduction technique which retrieves a set of basis probability distributions in order to minimise the reconstruction Kullback-Leibler (KL) divergence.

However, such compression methods require a discretisation of the probability density function and then its projection to a latent space, which in the case of E-PCA, requires solving an convex optimisation problem (iterative Newton's method) at each time step.

Furthermore, it is also not clear what effect an improved belief compression method would have on the policy. The better the compression method (in terms of KL divergence) the more dimensions are necessary and as a result more data points will be required to train the GMM. We make the observation in both Chapter 3 and Chapter 4 that the GMM policy is better than a myopic policy. However, this holds true only when a large amount of uncertainty is present. When the uncertainty starts to decrease the Greedy method performs just as well or even better than a four dimensional belief state GMM. As such, it is not clear that a more sophisticated belief compression method for the tasks we consider would be an improvement.

Policy representation

We learned the belief space policies in task space (Cartesian position in the world's frame of reference). This choice entails two difficulties: the number of parameters needed to encode the task and generalisation.

As there is much variance in the demonstrated search behaviour (at the raw trajectory level) many parameters are necessary to encode the policy. The

policies learned for the search tasks in this thesis required around 90 Gaussian functions of dimension 7. This is a considerable number of parameters considering the simplicity of the task (find the the table/wall, navigate to an edge, reach the goal). Typically more parameters also entail a poor generalisation.

Future work could be directed towards learning a high-level policy composed of parameterised behaviour policies which are easily re-usable. Such policies could be parameterised by sub-goals and contact constraints which could be extracted by segmenting the original demonstrations.

MLMF

The Measurement Likelihood Memory Filter (MLMF), introduced in Chapter 5, is based on the assumption of a sparse likelihood function where mostly zero measurements are present. This by itself is not a weakness, but a necessary assumption to achieve a linear space complexity. As the time complexity remains exponential with respect to the number of objects (although at a lesser growth rate) we were obliged to introduce an additional independence assumption. This assumption implies that some information will be lost and the filtered marginals will be an approximate solution with respect to an optimal Bayesian filter. There is no obvious remedy to this problem. There are two approaches: either introduce an assumption (as we did) or perform an approximate evaluation of the marginalisation. An approximate marginalisation could be achieved by evaluating the value of the marginals at specific points and setting the remaining marginal values by interpolation.

6.3 Final Words

During my PhD I have spent a considerable amount of time studying the role of uncertainty in Artificial Intelligence, specifically how it affects decision making in agents. I list below some points of interest and important insights.

- *Humans can be poor teachers:* Programming by Demonstration (PbD), traditionally requires an expert teacher to provide a set of demonstrations in the form of state-velocity pairs which are used to learn a policy represented by a regressor function. If the teacher is rarely successful at her/his task, learning a policy directly from her/his behaviour will yield a policy on par with the teachers performance. One of the original questions posed by PbD is *what to imitate ?* (Billard et al., 2008). By introducing a simple binary cost function, as shown in Chapter 4, we were able to improve the quality of the policy. All regression based PbD methods can use the Reinforcement Learning (RL) approach used in Chapter 4, with no additional rollouts being necessary.
- *Reinforcement Learning can be easy (continuous state and action space):*

RL is notable for needing lengthy simulations and episodes to be successful, which typically result in a complete exploration of the entire state (or parameter) space for tasks such as the inverted-pendulum or mountain cart. This is type of optimisation through random search is not feasible for learning a complicated continuous state and action space policy with a long time horizon.

In using RL during both my PhD and Master Thesis, there was always some magic required to get RL to work, such as choosing an appropriate learning rate of the value function and decay rate of the exploration noise, in order to get past local minimas.

There are two factors which make RL difficult to use: the exploration-exploitation dilemma and the non-stationarity of the value function approximator during on-line learning. To alleviate the exploration-exploitation problem one can use human demonstrations in which an optimal set of state-action pairs hopefully exists. The non-stationarity of the target value function can be achieved through batch methods, also known as fitted RL methods, which keep all data witnessed during episodes and learn the value function off-line through approximate dynamic programming. Learning the value function online at each time step can lead to divergence if an appropriate function approximator is not chosen (Szepesvári, 2010, p. 51). Given the current memory capacity of modern computers the fitted RL off-line methods seem more appropriate since the RL problem becomes a familiar regression problem for which many algorithms are applicable.

By using a fitted off-line approach to learn a value function in combination with a separate parameterisation of the policy in an Actor-Critic framework, it is again feasible to use simple reward functions which can result in significant improvements in the policy, as shown in Chapter 4.

Appendices

Appendix A

Peg in hole

A.1 Time to connect socket

We aim to test if there is any significant difference in time taken to connect socket A and B. We hypothesised that socket A requires longer time than socket B.

Data from two groups of 5 subjects where collected according the the experiment protocol. All subjects had time to familiarise themselves with the task and accomplished multiple training rounds before the start of the experiment. In Figure A.3 (*Top row*) we plot the time taken for each subject to connect the plug to the socket, once the socket was localised.

Before applying a statistical test to compare the time taken to connect the plug to the sockets for the two groups, we tested the normality of the time-distribution of each experiment condition (AA, AB, BA, BB).

We applied Shapiro-Wilk test of normality in R and used Q-Q plots to compare the shapes of distributions (Figure A.2) with normality. None of the time-distributions of the experimental conditions are normal ($p< 0.0001$). Therefore, we chose a non-parametric test to compare the distributions. Since socket A and B were performed by related samples we applied the Wilcoxon signed-rank test (a paired difference test). This test assesses whether the population mean rank differs between the two sockets (A and B) given that a subject performed both experiments one after the other.

Socket A took significantly longer time than socket B and this result was observed in the two groups (Group A $p<0.0001$, Group B $p=0.0002$, Wilcoxon signed-rank test, Figure A.3).

In summary, we observe that there is a significant effect of order. Starting with socket B greatly reduced the time taken for socket A ($p=0.02$). Regardless of the socket type between groups, the first socket always takes longer. For AA vs BA, AA is significantly longer ($p=0.02$, Wilcoxon rank sum test). For socket B, BB vs AB, BB takes significantly longer time ($p<0.0001$).

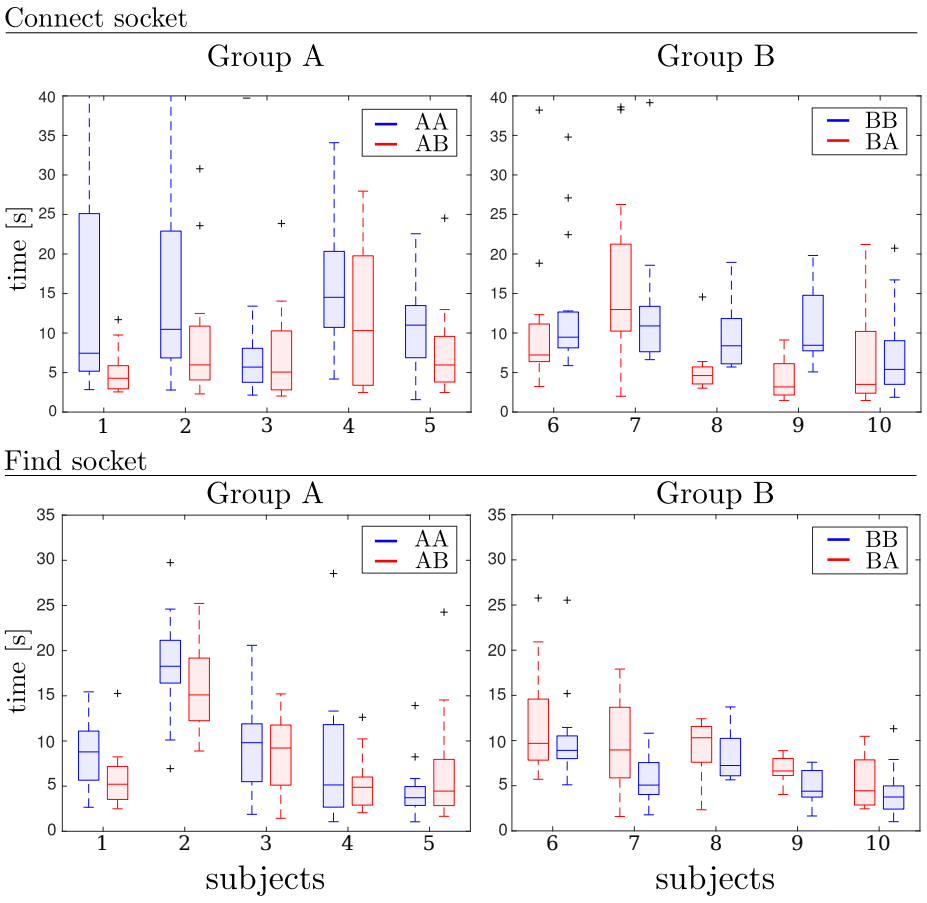


Figure A.1: Time taken to find and connect the plug to the power socket. **Top:** Time taken to connect the plug to the socket once the socket is localised. For most subjects the median value of the taken time is higher for socket A when compared with socket B. **Bottom**: time taken to localise the socket. For the second search, AB and BA, it seams that the subjects are faster indicating learning during the experiment.

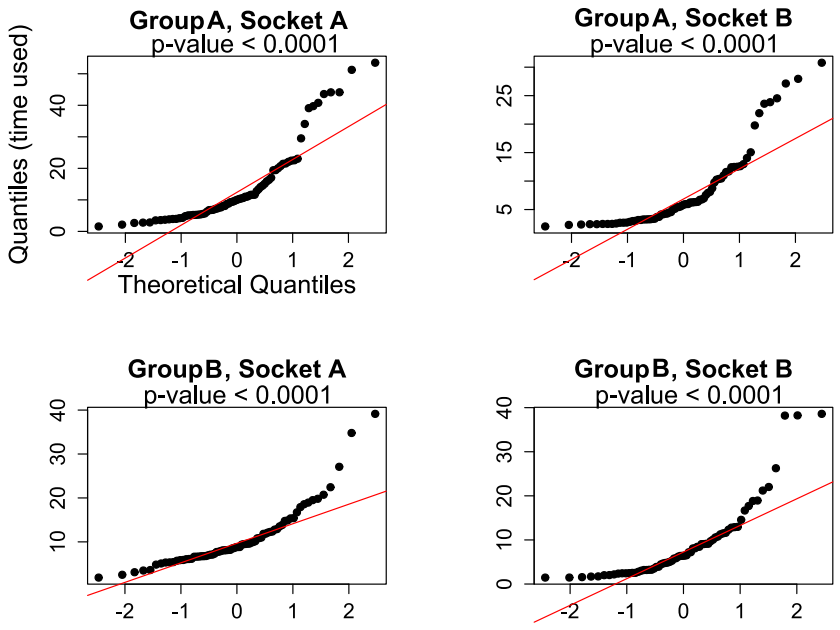


Figure A.2: Q-Q plots of time taken for four experiment conditions. The above p-values are from the Shapiro-Wilk test.

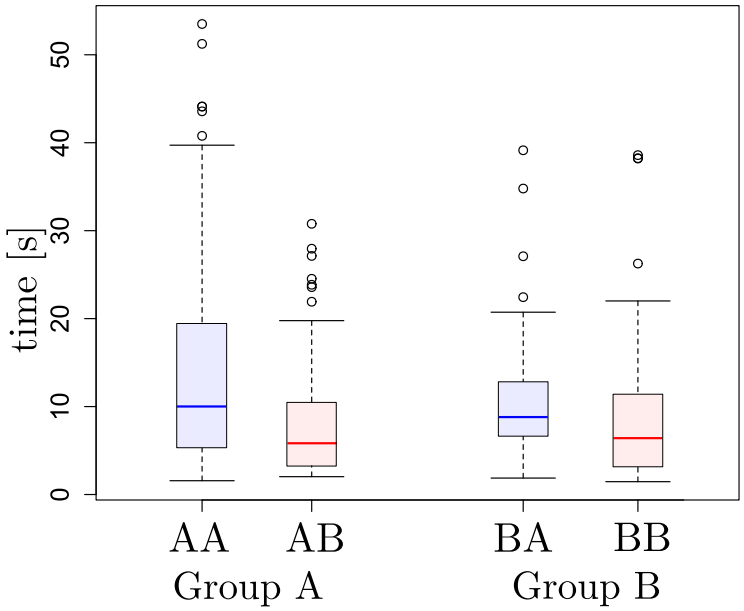


Figure A.3: Time taken to connect the sockets. For both groups, socket A always takes significantly longer time to connect, For Group A $p<0.0001$ and for Group B $p=0.0002$.

A.2 EM policy search

The objective of policy search in Reinforcement Learning (RL) is to optimise the parameters of a policy $\pi_{\boldsymbol{\theta}}$ (which is a Gaussian Mixture Model (GMM) in our case) such to maximise a cost function, $J(\boldsymbol{\theta})$ Equation A.2.1, defined over the parameters of the policy:

$$J(\boldsymbol{\theta}) = \overbrace{\sum_{i=1}^{N} \underbrace{\left(\prod_{t=0}^{T^{[i]}} \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \right)}_{p_{\boldsymbol{\theta}}(\tau_i)} R(\tau_i)}^{\mathbb{E}_{p_{\boldsymbol{\theta}}}\{R\}} \tag{A.2.1}$$

where $\tau_i = \{(\dot{x}_0, b_0), \cdots, (\dot{x}_T^{[i]}, b_T^{[i]})\}$ are the state-action samples of the ith episode. The total reward of one episode is the sum of discounted rewards:

$$R(\tau_i) = \sum_{t=0}^{\mathrm{T}^{[\mathrm{i}]}} \gamma^t \, r(b_t^{[i]}, \dot{x}_t^{[i]}) \tag{A.2.2}$$

where $\gamma \in [0, 1)$ is the discount factor which dictates how much future rewards are considered important at the current decision point. Most policy search methods are based gradient ascent on the cost function:

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta} + \alpha \, \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{A.2.3}$$

The drawback of policy gradient methods is that a learning rate, α, needs to be specified and the optimisation strongly depends on the fine tuning of this value. An alternative, which avoids the learning rate problem, are Expectation-Maximisation (EM) methods. As our policy is a Gaussian Mixture Model (GMM) we can take advantage of the already existing EM updates of the GMM which are well known. Also updating the covariance parameters of the GMM by gradient ascent can lead to problems such the covariances being no longer positive-definite and this resolution requires the addition of constraints which becomes cumbersome. In contrast the EM solution for the GMM case is simple and easy to implement.

EM policy search methods have two phases. The first phase consists of generating a set of episodes $\boldsymbol{\tau}$ from the policy, this is the E-step. The second step consists of finding a new parameter $\boldsymbol{\theta}^{\text{new}}$ of the policy by maximising the cost function given the episodes:

$$\boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, J(\boldsymbol{\theta}) \tag{A.2.4}$$

The above optimisation can be solved through the same approach taken in EM which consists of maximising the the logarithmic lower bound of the cost function $J(\boldsymbol{\theta})$:

$$\log(J(\boldsymbol{\theta})) = \log \sum_{i=1}^{N} \frac{p_{\boldsymbol{\theta}}(\tau_i)}{p_{\boldsymbol{\theta}'}(\tau_i)} \, p_{\boldsymbol{\theta}'}(\tau_i) \, R(\tau_i)$$

$$\geq \underbrace{\sum_{i=1}^{N} \log \left(\frac{p_{\boldsymbol{\theta}}(\tau_i)}{p_{\boldsymbol{\theta}'}(\tau_i)} \right) p_{\boldsymbol{\theta}'}(\tau_i) \, R(\tau_i)}_{\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}')} \tag{A.2.5}$$

The parameter $\boldsymbol{\theta}'$ belongs to the policy used to generate the episodes in the E-step and the parameter $\boldsymbol{\theta}$ is the one we are optimising for. The lower bound $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ is derived from Jensen's inequality. The lower bound is next maximised by taking its derivative $\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = 0$.

$$\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}}(\tau_i) \right) p_{\boldsymbol{\theta}'}(\tau_i) \, R(\tau_i) -$$

$$\underbrace{\nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}'}(\tau_i) \right)}_{=0} p_{\boldsymbol{\theta}'}(\tau_i) \, R(\tau_i) \tag{A.2.6}$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbb{E}_{p_{\boldsymbol{\theta}'}} \Big\{ \nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}}(\tau_i) \right) R(\tau_i) \Big\} \tag{A.2.7}$$

$$= \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}} \log \left(\pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \right) R(\tau_i) \tag{A.2.8}$$

$$= \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \, Q^{\pi_{\boldsymbol{\theta}'}}(\dot{x}_t^{[i]}, b_t^{[i]}) \tag{A.2.9}$$

From A.2.7 to A.2.8 we used the property: $\log(\prod \pi_{\boldsymbol{\theta}}) = \sum \log(\pi_{\boldsymbol{\theta}})$ and the expectation vanishes as we evaluated it through sampling from the policy $\pi_{\boldsymbol{\theta}'}$ (E-step). From A.2.8 to A.2.9 we used the fact that previous discounted rewards do not dependent on future time steps. The reader is referred to (Deisenroth et al., 2013, p. 50) for more details regarding Expectation-Maximisation and policy search in reinforcement learning. In the next section A.3 we maximise A.2.9 for the case when the policy is parameterised by a GMM.

A.3 Q-EM for GMM

We derive the EM update rules of the lower bound $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ for a policy $\pi_{\boldsymbol{\theta}}(\dot{x}, b)$ parameterised by a Gaussian Mixture Model which we call Q-EM. Below we redefine the GMM function for convenience:

$$\pi_{\boldsymbol{\theta}}(\dot{x}, b) = \sum_{k=1}^{K} w^{[k]} \, g(\dot{x}, b; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}) \tag{A.3.1}$$

The parameters $\boldsymbol{\theta} = \{w^{[k]}, \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{1,\ldots,K}$, are the weights, means and covariances of the individual Gaussian functions, $g(\cdot)$.

Finding the new parameters of the GMM given a cost function $J(\boldsymbol{\theta})$ consists of maximising its logarithmic lower bound, Equation A.3.2:

$$\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\dot{x}_t^{[i]}, b_t^{[i]}) \tag{A.3.2}$$

As the ordering of the episode samples in the above equation does not mater we concatenate all the episodes into one dataset $\mathcal{D}$ and the jth sample in this dataset is given by $\mathbf{x}^{[j]} = [\dot{x}^{[j]}, b^{[j]}]^{\mathrm{T}}$.

$$\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}^{[j]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]}) \tag{A.3.3}$$

To maximise Equation A.3.3 we take the derivatives with respect to the parameters $\boldsymbol{\theta} = \{w, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ of the GMM. The reader my notice that the above equation is the derivative of the log-likelihood of the dataset $\mathcal{D}$ weighted by $Q^{\pi_{\boldsymbol{\theta}'}}$, a constant scalar value. As a consequence the result of maximisation will be similar to the standard EM solution of the GMM with an additional weighting factor. For a reference on the derivative of the log-likelihood of a GMM the reader is referred to (Bishop, 2006, Chap. 9).

Q-EM maximisation

$$\nabla_{\boldsymbol{\mu}^{[k]}} \mathcal{Q}(\boldsymbol{\mu}^{[k]}, \boldsymbol{\theta}') = \sum_{j=1}^{M} \underbrace{\frac{w^{[k]}\, g(\mathbf{x}^{[j]}; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]})}{\sum_{l=1}^{M} w^{[l]}\, g(\mathbf{x}; \boldsymbol{\mu}^{[l]}, \boldsymbol{\Sigma}^{[l]})}}_{\gamma_k(\mathbf{x}^{[j]})} {\boldsymbol{\Sigma}^{[k]}}^{-1} (\mathbf{x}^{[j]} - \boldsymbol{\mu}^{[k]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]}) = 0 \tag{A.3.4}$$

In A.3.4 we used the same notation and derivation as in (Bishop, 2006, Chap. 9.2.2), where $\gamma_k(\mathbf{x}^{[j]})$ is the responsibility factor, denoting the probability that data point $\mathbf{x}^{[j]}$ belongs to Gaussian function k. After rearrangement the new mean is given by Equation A.3.5.

$$\boldsymbol{\mu}_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} \gamma_k(\mathbf{x}^{[j]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})\, \mathbf{x}^{[j]}}{\sum_{j=1}^{M} \gamma_k(\mathbf{x}^{[j]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})} \tag{A.3.5}$$

$$\boldsymbol{\Sigma}_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} \gamma_k(\mathbf{x}^{[j]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})(\mathbf{x}^{[j]} - \boldsymbol{\mu}_{\mathbf{new}}^{[k]})(\mathbf{x}^{[j]} - \boldsymbol{\mu}_{\mathbf{new}}^{[k]})^{\mathrm{T}}}{\sum_{j=1}^{M} \gamma_k(\mathbf{x}^{[j]})\, Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})} \tag{A.3.6}$$

$$w_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})\, \gamma_k(\mathbf{x}^{[j]})}{\sum_{j=1}^{M} Q^{\pi_{\boldsymbol{\theta}'}}(\mathbf{x}^{[j]})} \tag{A.3.7}$$

The covariances, Equation A.3.6, and weights, Equation A.3.7, are derived in a similar fashion.

A.4 Unbiased estimator

The temporal difference error is an unbiased estimate of the advantage function:

$$\begin{aligned} \mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}}\{\delta_t^{\pi} | b_t, u_t\} &= \mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}}\{r_{t+1} + \gamma V^{\pi}(b_{t+1}) | b_t, u_t\} - V^{\pi}(b_t) \\ &= Q^{\pi}(b_t, u_t) - V^{\pi}(b_t) \\ &= A^{\pi}(b_t, u_t) \end{aligned} \tag{A.4.1}$$

Appendix B

Non-parametric Bayesian State Space Estimator

B.1 Probabilities

There are two rules extensively used in the derivation of a Bayesian filter recursion regardless of the chosen parameterisation. Although well known we restate them here for convenience.

- **Chain rule**

$$P(X_M, \cdots, X_1) = P(X_M|X_{M-1}, \cdots, X_1)P(X_{M-1}, \cdots, X_1) \tag{B.1.1}$$

- **Conditional independence**

$$\begin{aligned} P(A, B|C) &= P(A|\cancel{B}, C)P(B|C) \\ &= P(A|C)P(B|C) \end{aligned} \tag{B.1.2}$$

A and B are independent given that C is known.

B.2 Bayesian filtering recursion

Joint distribution

The joint distribution is over all the random variables since $t = 0$ until the current time step t:

$$P(A_{0:t}, O, Y_{0:t}|u_{1:t}) =$$

$$P(A_0)P(O)P(Y_0|A_0, O)\prod_{t=1}^{t} P(A_t|A_{t-1}, u_t)P(Y_t|A_t, O) \tag{B.2.1}$$

$$P(A_0)P(O)P(Y_0|A_0, O)P(A_{1:t}|u_{1:t})P(Y_{1:t}|A_{1:t}, O) \tag{B.2.2}$$

$$P(A_0)P(O)P(A_{1:t}|u_{1:t})P(Y_{0:t}|A_{0:t}, O) \tag{B.2.3}$$

From Equation B.2.1 to B.2.2 we made use of the chain rule of probabilities

(Equation B.1.1) and the conditional independence $A_{t+1} \perp\!\!\!\perp A_{t-1}|A_t$, see below a more concrete example for $P(A_{0:3}|u_{1:3})$:

$$
\begin{aligned}
&P(A_3, A_2, A_1, A_0|u_{1:3}) = \\
&P(A_3|A_2, \cancel{A_1}, \cancel{A_0}, u_{1:3})P(A_2|A_1, \cancel{A_0}, u_{1:3})P(A_1|A_0, u_{1:3})P(A_0|u_{1:3}) = && \text{(B.2.4)}\\
&P(A_3|A_2, u_3, \cancel{u_{1:2}})P(A_2|A_1, u_2, \cancel{u_{1:3}})P(A_1|A_0, u_1, \cancel{u_{2:3}})P(A_0|\cancel{u_{1:3}}) = && \text{(B.2.5)}\\
&\underbrace{(A_3|A_2, u_3)P(A_2|A_1, u_2)P(A_1|A_0, u_1)}_{P(A_{1:3}|u_{1:3})}P(A_0) && \text{(B.2.6)}
\end{aligned}
$$

We applied the chain rule to get Equation B.2.4 and the cancellations arise from conditional independence between the agent random variable A_t specified by the BN, see Figure 5.3 on page 127. The cancellations on line B.2.5 arise also from BN, A_t only depends on u_t and no previous actions.

To see clearer the relationship between the left and right hand side of Equation B.2.3, we can again apply the chain rule to the right, which gives:

$$
\begin{aligned}
P(A_{0:t}, O, Y_{0:t}|u_{1:t}) &= P(Y_{0:t}|A_{0:t}, O, \cancel{u_{1:t}})P(A_{0:t}, O|u_{1:t}) && \text{(B.2.7)}\\
&= P(Y_{0:t}|A_{0:t}, O)P(A_{0:t}, O|u_{1:t}) && \text{(B.2.8)}
\end{aligned}
$$

The measurements $Y_{0:t}$ are independent of the actions $u_{0:t}$ given the history of the agent's random variables $A_{0:t}$. Next we see the relation between the thirst three terms on the right of Equation B.2.3 with $P(A_{0:t}, O|u_{1:t})$. We apply the chain rule:

$$
\begin{aligned}
P(A_{0:t}, O|u_{1:t}) &= P(A_{0:t}|\cancel{O}, u_{1:t})P(O|\cancel{u_{1:t}}) && \text{(B.2.9)}\\
&= P(A_0)P(O)P(A_{1:t}|u_{1:t}) && \text{(B.2.10)}
\end{aligned}
$$

The object O is conditionally independent of the actions and the second of actions $A_{0:t}$ are conditional independent of the object as they are dependent only on the measurements.

Typically we are interested in the filtered joint distribution $P(A_t, O, Y_{0:t}|u_{1:t})$ which is obtained by marginalising over $A_{0:t-1}$:

$$
P(A_t, O, Y_{0:t}|u_{1:t}) = \sum_{A_{0:t-1}} P(A_t, A_{0:t-1}, O, Y_{0:t}|u_{1:t}) \qquad \text{(B.2.11)}
$$

A recursion exists making it not necessary to sum over all agents random variables from the state time until then end, but instead we only have to consider the last time step, see the next paragraph.

Filtering problem

We derive $P(A_t, O, Y_{0:t}|u_{1:t})$, we start from the joint distribution, Equation

B.2.12:

$$P(A_t, O, Y_{0:t}|u_{1:t}) = \sum_{A_{t-1}} P(A_t, A_{t-1}, O, Y_t, Y_{0:t-1}|u_t, u_{1:t-1}) \tag{B.2.12}$$

$$\begin{aligned} P(A_t, O, Y_{0:t}|u_{1:t}) \quad = \textstyle\sum_{A_{t-1}} \quad & P(Y_t|\cancel{Y_{0:t-1}}, A_t, \cancel{A_{t-1}}, O, \cancel{u_t}, \cancel{u_{1:t-1}}) \\ & P(A_t|A_{t-1}, \cancel{O}, u_t, \cancel{Y_{0:t-1}}, \cancel{u_{1:t-1}}) \\ & P(A_{t-1}, O, Y_{0:t-1}|\cancel{u_t}, u_{1:t-1}) \end{aligned}$$

$$\begin{aligned} & P(A_t, O, Y_{0:t}|u_{1:t}) \qquad \text{(B.2.13)} \\ & = \sum_{A_{t-1}} P(Y_t|A_t, O) P(A_t|A_{t-1}, u_t) P(A_{t-1}, O, Y_{0:t-1}|u_{1:t-1}) \\ & = P(Y_t|A_t, O) \underbrace{\sum_{A_{t-1}} P(A_t|A_{t-1}, u_t) P(A_{t-1}, O, Y_{0:t-1}|u_{1:t-1})}_{P(A_t, O, Y_{0:t-1}|u_{1:t})} \qquad \text{(B.2.14)} \end{aligned}$$

$$\begin{aligned} P(A_t, O, Y_{0:t}|u_{1:t}) &= P(Y_t|A_t, O) P(A_t, O, Y_{0:t-1}|u_{1:t}) \\ &= P(Y_t|A_t, O) P(A_t, O|Y_{0:t-1}, u_{1:t}) P(Y_{0:t-1}|u_{1:t}) \end{aligned} \tag{B.2.15}$$

All the cancellations come from the *Markov Assumption* read from the structure of the Bayesian network. The resulting final Bayesian recursion is obtained by conditioning on the measurement and actions, which is the normalisation factor.

$$\begin{aligned} P(A_t, O|Y_{0:t}, u_{1:t}) &= \frac{P(Y_t|A_t, O) P(A_t, O|Y_{0:t-1}, u_{1:t}) P(Y_{0:t-1}|u_{1:t})}{P(Y_{0:t}|u_{1:t})} \\ &= \frac{P(Y_t|A_t, O) P(A_t, O|Y_{0:t-1}, u_{1:t}) \cancel{P(Y_{0:t-1}, u_{1:t})}}{P(Y_t|Y_{0:t-1}, u_{1:t}) \cancel{P(Y_{0:t-1}|u_{1:t})}} \end{aligned} \tag{B.2.16}$$

$$P(A_t, O|Y_{0:t}, u_{1:t}) = \frac{P(Y_t|A_t, O) P(A_t, O|Y_{0:t-1}, u_{1:t})}{P(Y_t|Y_{0:t-1}, u_{1:t})} \tag{B.2.17}$$

The evidence is the the integration of all terms which are not measurements in the numerator of Equation B.2.17.

$$P(Y_t|Y_{0:t-1}, u_{1:t}) = \sum_{A_t} \sum_{O} P(Y_t|A_t, O) P(A_t, O|Y_{0:t-1}, u_{1:t}) \tag{B.2.18}$$

$$= \sum_{A_t} \sum_{O} P(Y_t, A_t, O|Y_{0:t-1}, u_{1:t}) \tag{B.2.19}$$

This is very expensive since we have to sum over the entire joint distribution, the MLMF avoids doing this by only considering the dependent regions of the joint distribution.

B.3 Recursion example

Derivation of the filtered joint distribution, $P(A_t, O, Y_t | Y_{0:t}, u_{1:t})$, for two updates. At initialisation when no action has yet been taken the filtered joint distribution is the product of the initial marginals and first likelihood function:

$$P(A_0, O, Y_0) = P(O)P(A_0)P(Y_0|A_0, O) \tag{B.3.1}$$

The a first action, u_1 is applied, which to get the filtered joint distribution is marginalised:

$$\begin{aligned}
P(A_1, O, Y_0|u_1) &= P(O) \sum_{A_0} P(A_1|A_0, u_1)P(A_0)P(Y_0|A_0, O) && \text{(B.3.2)}\\
&= P(O) \sum_{A_0} P(A_1, A_0, Y_0|u_1, O) && \text{(B.3.3)}\\
&= P(O)P(A_1, Y_0|u_1, O) && \text{(B.3.4)}\\
&= P(O)P(Y_0|A_1, O, u_1)P(A_1|u_1, \cancel{O}) && \text{(B.3.5)}\\
&= P(O)P(Y_0|A_1, O, u_1)P(A_1|u_1) && \text{(B.3.6)}
\end{aligned}$$

From Equation B.3.4 to B.3.5 we used the Chain rule B.1.1 and the cancellation in Equation B.3.5 arise from the factorisation of the joint distribution, see Figure 5.3 on page 127, A's marginal does not depend on O. After the application of the first action, the filtered joint has the following form:

$$P(A_1, O, Y_0|u_1) = P(O)P(A_1|u_1)P(Y_0|A_1, O, u_1) \tag{B.3.7}$$

A second measurement Y_1 and action u_2 are integrated into the filtered joint distribution:

$$\begin{aligned}
&P(A_2, O, Y_{0:1}|u_{1:2}) = \\
&P(O) \sum_{A_1} P(A_2|A_1, u_2)P(A_1|u_1)P(Y_0|A_1, O, u_1)P(Y_1|A_1, O) = \\
&P(O) \sum_{A_1} P(A_2, A_1|u_{1:2})P(Y_{0:1}|A_1, O, u_1) = \\
&P(O) \sum_{A_1} P(A_2, A_1, Y_{0:1}|O, u_{1:2}) = \\
&P(O)P(A_2, Y_{0:1}|O, u_{1:2}) = && \text{(B.3.8)}\\
&P(O)P(Y_{0:1}|A_2, O, u_{1:2})P(A_2|\cancel{O}, u_{1:2}) && \text{(B.3.9)}
\end{aligned}$$

We expand the function $P(Y_{0:1}|A_2, O, u_{1:2})$ to give a sense of how the likelihood function's positions get as illustrated in Figure 5.6 on page 132.

$$P(Y_0, Y_1|A_2, O, u_1, u_2) = P(Y_0|\cancel{Y_1}, A_2, O, u_1, u_2)P(Y_1|A_2, O, \cancel{u_1}, u_2) \quad \text{(B.3.10)}$$
$$= P(Y_0|A_2, O, u_{1:2})P(Y_1|A_2, O, u_2) \quad \text{(B.3.11)}$$

The first likelihood of measurement Y_0 is dependent on the last to applied actions whilst the likelihood of Y_1 is dependent on the last action.

Repeating the above for $Y_{2:t}$ and $u_{3:t}$ results in:

$$P(A_t, O, Y_{0:t}|u_{1:t}) = P(O)P(A_t|u_{1:t})\prod_{i=0}^{t} P(Y_i|A_t, O, u_{i+1:t}) \quad \text{(B.3.12)}$$

If $t = 3$, ($Y_{0:3}$ and $u_{1:3}$) according to the above equation we would get:

$$\begin{aligned} P(A_3, O, Y_{0:3}|u_{1:3}) = P(O)P(A_3|u_{1:3})&P(Y_0|A_3, O, u_{1:3}) \\ &P(Y_1|A_3, O, u_{2:3}) \\ &P(Y_2|A_3, O, u_{3:3}) \\ &P(Y_3|A_3, O, \cancel{u_{4:3}}) \end{aligned} \quad \text{(B.3.13)}$$

We introduce some notation rules, first if $(i+1) > t$ for $u_{(i+1):t}$ then it cancels out since the current measurement Y_t cannot depend on a future action $u_{(i+1)}$.

B.4 Derivation of the evidence

The evidence, also known as the marginal likelihood, is the marginalisation of all non measurement random variables from the filtered joint distribution $P(A_t, O, Y_{0:t}|u_{1:t})$. We detail below how we compute the evidence in a recursive manner whilst only considering dependent regions of the joint distribution.

We start with the **standard** definition of the evidence:

$$P(Y_{0:t}|u_{1:t}) = \sum_{A_t}\sum_{O} P(A_t, O, Y_{0:t}|u_{1:t}) \in \mathbb{R} \quad \text{(B.4.1)}$$

If both A_t and O are random variables defined over a discretised state space of N states, the above double integral will sum a total of N^2 states which is the complete state space of the joint distribution $P(A_t, O, Y_{0:t}|u_{1:t}) \propto P(A_t, O|Y_{0:t}, u_{1:t})$, see Figure 5.7 on page 136 for an illustrate of such a joint distribution. As we are interested in a recursive computation of the evidence, we consider the gradient:

$$\alpha_t = \nabla_{Y_t} P(Y_{0:t}|u_{1:t}) = P(Y_{0:t}|u_{1:t}) - P(Y_{0:t-1}|u_{1:t}) \quad \text{(B.4.2)}$$

$$\alpha_t = \sum_{A_t} \sum_{O} P(A_t, O, Y_{0:t}|u_{1:t}) - P(A_t, O, Y_{0:t-1}|u_{1:t}) \tag{B.4.3}$$

$$= \sum_{A_t} \sum_{O} P(Y_t|A_t, O) P(A_t, O, Y_{0:t-1}|u_{1:t}) - P(A_t, O, Y_{0:t-1}|u_{1:t}) \tag{B.4.4}$$

$$= \sum_{A_t} \sum_{O} (P(Y_t|A_t, O) - 1) P(A_t, O, Y_{0:t-1}|u_{1:t}) \tag{B.4.5}$$

The gradient α_t is the difference in mass before and after the application the likelihood function, $P(Y_t|A_t, O)$. The above summation, Equation B.4.5, is over the entire joint distribution state space. We can take advantage of the fact that the likelihood function is sparse and will only affect a small region of the joint distribution, which we called the dependent states, $\cap$. The states which are not affected by the joint distribution will result in a contribution of zero to Equation B.4.5. We rewrite the gradient update in terms of only the dependent regions:

$$\alpha_t = \sum_{A_t} \sum_{O} (P(Y_t|A_t, O) - 1) P_{\cap}(A_t, O, Y_{0:t-1}|u_{1:t}) \tag{B.4.6}$$

Consider the first update of the evidence at time $t = 0$:

$$\alpha_0 = \sum_{A_t} \sum_{O} (P(Y_0|A_0, O) - 1) P(A_0, O) \tag{B.4.7}$$

The one in Equation B.4.8 is the original value of the normalisation denominator before any observation is made and as the initial joint distribution $P(A_0, O)$ is normalised the value of the denominator is one.

$$P(Y_0) = 1 + \alpha_0 \tag{B.4.8}$$

For the evidence $P(Y_{0:t}|u_{1:t})$ we consider the summation of all the derivatives α_t from time $t = 0$ until t:

$$P(Y_{0:t}|u_{1:t}) = 1 + \sum_{t=0}^{t} \alpha_t \tag{B.4.9}$$

B.5 Derivation of the marginal

The marginal of a random variable is the marginalisation or integration over all other random variables, $P(A_t, |Y_{0:t}) = \sum_{O} P(A_t, O|Y_{0:t})$. Below we give a form of this integration which exploits the independent regions in the joint distribution.

$$P(A_t, |Y_{0:t}) = \mathbf{P(A_t|Y_{0:t-1})} - \Big(\mathbf{P(A_t|Y_{0:t-1})} - P(A_t|Y_{0:t})\Big) \tag{B.5.1}$$

In Equation B.5.1 we add and substract $P(A_t|Y_{0:t-1})$ and we further split $P(A_t|Y_{0:t-1})$ into independent and dependent components:

$$\begin{aligned} &P(A_t, |Y_{0:t}) = P(A_t|Y_{0:t-1}) - \\ &\Big(\underbrace{P_{\cap}(A_t|Y_{0:t-1}) + \cancel{P_{\ominus}(A_t|Y_{0:t-1})}}_{P(A_t|Y_{0:t-1})} - \underbrace{P_{\cap}(A_t|Y_{0:t}) + \cancel{P_{\ominus}(A_t|Y_{0:t})}}_{P(A_t|Y_{0:t})}\Big) \end{aligned} \tag{B.5.2}$$

From equation B.5.2 to B.5.3 we used the fact that independent regions of the marginal distributions will remain unchanged after an observation, $P_{\ominus}(A_t|Y_{0:t-1}) = P_{\ominus}(A_t|Y_{0:t})$, and before re-normalisation. This results in the final recursive update:

$$P(A_t, |Y_{0:t}) = P(A_t|Y_{0:t-1}) - \Big(P_{\cap}(A_t|Y_{0:t-1}) - P_{\cap}(A_t|Y_{0:t})\Big) \tag{B.5.3}$$

Equation B.5.3 states that only elements of the marginals which are dependent will change by the difference before and after a measurement update.

B.6 MLMF motion and measurement update

In Algorithm 3 we detail the motion-measurement update and initialisation steps of the MLMF (the parameters of $P_{\cap}(A_t, O, Y_{0:t-1}|u_{1:t})$ and the corresponding marginals are not shown). At initialisation the parameters of the filtered marginals are set equal to the marginals of the MLMF joint distribution. In the motion step the agent's marginals of the filtered and joint distribution are updated in accordance with the motion model. The offsets $l_{0:t}$ of the memory likelihood function get the current action added to them. In the measurement step the evidence is computed, the filtered marginals are updated by only carrying out the marginalisation in the dependent states of the joint distribution. Finally the current measurement Y_t is added to the memory likelihood's parameters with $l_t = 0$. This formulation is advantageous as the joint distribution is only evaluated inside the dependent regions $A \cap O$ of the joint distribution. All the parameters $\Psi_{0:t}$ of the memory likelihood function $P(Y_{0:t}|A_t, O, u_{1:t}; \Psi_{0:t})$ are retained. We keep track of the normalisation factor, the evidence $P(Y_{0:t}|u_{1:t}; \alpha_{0:t})$ which is a scalar quantity. With these parameters it is possible to reconstruct the joint distribution, however the MLMF-SLAM algorithm only makes changes to the filter marginals, see Figure 5.9. We evaluate this formulation of the joint distribution with the standard histogram filter in the case of the 1D filtering scenario illustrated in Figure 5.6 on page 132 and

we find them to be identical. Having respected the formulation of Bayes rule, we assert that Algorithm 3 is a Bayesian Optimal Filter[1].

[1] An optimal Bayesian solution is an exact solution to the recursive problem of calculating the exact posterior density Arulampalam et al. (2002)

Algorithm 3: MLMF-SLAM

input :

measurements

$\mathbf{Y_t}$, $\mathbf{u_t}$

joint parameters:

$P(A_{t-1}|u_{1:t-1};\boldsymbol{\theta}_a^*)$ $P(O;\boldsymbol{\theta}_o^*)$, $\Psi_{0:t-1}$, $\alpha_{0:t-1}$

filtered marginals:

$P(A_{t-1}|Y_{0:t-1},u_{1:t-1};\boldsymbol{\theta}_a)$, $P(O|Y_{0:t-1};\boldsymbol{\theta}_o)$

output:

joint parameters:

$P(A_t|u_{1:t};\boldsymbol{\theta}_a^*)$, $\Psi_{0:t}$, $\alpha_{0:t}$

filtered marginals:

$P(A_t|Y_{0:t},u_{1:t};\boldsymbol{\theta}_a)$, $P(O|Y_{0:t};\boldsymbol{\theta}_o)$

---

initialisation

$$P(A_0;\boldsymbol{\theta}_a) := P(A_0;\boldsymbol{\theta}_a^*)$$
$$P(O;\boldsymbol{\theta}_o) := P(O;\boldsymbol{\theta}_o^*)$$
$$\Psi_0 := \{\}$$
$$\alpha_0 := 0$$

---

motion update

$$P(A_t|u_{1:t};\boldsymbol{\theta}_a^*) = \sum_{A_{t-1}} P(A_t|A_{t-1},\mathbf{u_t})P(A_{t-1}|u_{1:t-1};\boldsymbol{\theta}_a^*)$$
$$P(A_t|Y_{0:t-1},u_{1:t};\boldsymbol{\theta}_a) = \sum_{A_{t-1}} P(A_t|A_{t-1},\mathbf{u_t})P(A_{t-1}|Y_{0:t-1},u_{1:t-1};\boldsymbol{\theta}_a)$$
$$\bar{\Psi}_{0:t} \leftarrow \Psi_{0:t-1}(\mathbf{u_t}) : \text{Algorithm 2 } \textit{(motion update)}$$

---

measurement update

$$\alpha_{0:t} = \alpha_{0:t-1} + \sum_{A_t}\sum_{O}\Big(P(\mathbf{Y_t}|A_t,O) - 1\Big)P_\cap(A_t,O,Y_{0:t-1}|u_{1:t})$$
$$P(Y_{0:t}|u_{1:t};\boldsymbol{\alpha_{0:t}}) = 1 + \alpha_{0:t}$$
$$P(A_t|Y_{0:t};\boldsymbol{\theta}_a) = P(A_t|Y_{0:t-1};\boldsymbol{\theta}_a) - \Big(P_\cap(A_t|Y_{0:t-1}) - P_\cap(A_t|Y_{0:\mathbf{t}})\Big)$$
$$P(O|Y_{0:t};\boldsymbol{\theta}_o) = P(O|Y_{0:t-1};\boldsymbol{\theta}_o) - \Big(P_\cap(O_t|Y_{0:t-1}) - P_\cap(O_t|Y_{0:\mathbf{t}})\Big)$$
$$\Psi_{0:t} \leftarrow \bar{\Psi}_{0:t}(\mathbf{Y_t}) : \text{Algorithm 2 } \textit{(measurement update)}$$

B.7 Scalabe-MLMF Algorithm

To address exponential time complexity a an independence assumption was introduced between the objects. This leads to a new filtering algorithm in which each agent-object joint distribution pair $P(A_t^{(i)}, O^{(i)}|Y_{0:t}^{(i)}, u_{1:t})$ is filtered independently of one another until a positive contact has been sensed. Then the exchange of information of one joint distribution to another is achieved through the agent's marginals $A^{(i)}$ according to Algorithm 4. The measurement update is the same as previously described in Algorithm 3 in the case of no positive measurements of the objects. If the agent senses an object, all of the agent marginals of the remaining joint distributions are set to the marginal of the joint distribution pair belonging to the positive measurement $Y_t^{(i)}$.

Algorithm 4: Scalable-MLMF: Measurement Update

input : $P(A_t^{(i)}|u_{1:t})$, $P(A_t^{(i)}|Y_{0:t-1}^{(i)}, u_{1:t})$
$P(O^{(i)})$, $P(O^{(i)}|Y_{0:t-1}^{(i)}, u_{1:t})$
$Y_t^{(i)}$
$i = 1, \cdots, M$

▷ `If object` i `has been sensed by the agent`

if $Y_t^{(i)} == 1$ **then**
  $P(O^{(i)}|Y_{0:t}^{(i)}) \leftarrow P(O^{(i)}|Y_{0:t-1}^{(i)})$; ▷ `measurement update Algo.` 3
  $P(A_t^{(i)}|Y_{0:t}^{(i)}, u_{1:t}) \leftarrow P(A_t^{(i)}|Y_{0:t-1}^{(i)}, u_{1:t})$
  forall $j \in (1, \dots M-1) \setminus i$ **do**
    $P(A_t^{(j)}|Y_{0:t}, u_{1:t}) = P(A_t^{(i)}|Y_{0:t}, u_{1:t})$
    $P(A_t^{(j)}|u_{1:t}) = P(A_t^{(i)}|u_{1:t})$
    $P(O^{(j)}|Y_{0:t}^{(i)}) \leftarrow \sum_{A^{(j)}} P(A_t^{(j)}, O^{(j)}|Y_{0:t}^{(i)})$
else
  forall $i \in (1, \dots M)$ **do**
    measurement update Algo. 3

References

Douglas Aberdeen and Jonathan Baxter. Scaling internal-state policy-gradient methods for pomdps. In *International Conference on Machine Learning (ICML)*, pages 3–10, 2002. URL http://users.rsise.anu.edu.au/~daa/files/papers/gradIstate-icml.pdf.

Fares Abu-Dakka, Bojan Nemec, Aljaz Kramberger, Anders Glent Buch, Norbert Krüger, and Ales Ude. Solving peg-in-hole tasks by human demonstration and exception strategies. *Industrial Robot*, 41(6):575–584, 2014. URL http://dx.doi.org/10.1108/IR-07-2014-0363.

Alejandro Agostini and Enric Celaya. Reinforcement learning with a gaussian mixture model. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010. URL http://dx.doi.org/10.1109/IJCNN.2010.5596306.

Ali akbar Agha-mohammadi, Suman Chakravorty, and Nancy Amato. Firm: Feedback controller-based information-state roadmap - a framework for motion planning under uncertainty. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4284–4291, Sept 2011. URL http://dx.doi.org/10.1109/IROS.2011.6095010.

Ali akbar Agha-mohammadi, Saurav Agarwal, Aditya Mahadevan, Suman Chakravorty, Daniel Tomkins, Jory Denny, and Nancy Amato. Robust online belief space planning in changing environments: Application to physical mobile robots. In *International Conference on Robotics and Automation (ICRA)*, pages 149–156, May 2014. URL http://dx.doi.org/10.1109/ICRA.2014.6906602.

Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Transactions on Signal Processing*, 50(2):174–188, Feb 2002. URL http://dx.doi.org/10.1109/78.978374.

Christopher Atkeson, Andrew Moore, and Stefan Schaal. Locally weighted learning. *Artificial Inteligence*, pages 11–73, 1997. URL http://dx.doi.org/10.1023/A:1006559212014.

Chris Baker, Joshua. Tenenbaum, and Rebecca Saxe. Bayesian models of human action understanding. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 99–106. 2006. URL http://papers.nips.cc/paper/2815-bayesian-models-of-human-action-understanding.pdf.

Chris Baker, Joshua Tenenbaum, and Rebecca Saxe. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Journal of Cognitive Science*, 2011. URL https://mindmodeling.org/cogsci2011/.

Simon Baron-Cohen. *Mindblindness*. 1995.

Jonathan Baxter and Peter Bartlett. Reinforcement learning in pomdp's via direct gradient ascent. In *International Conference on Machine Learning (ICLM)*, pages 41–48. Morgan Kaufmann, 2000.

Mohamad Bdiwi, Alexander Winkler, Michael Jokesch, and Jozef Suchy. Improved peg-in-hole (5-pin plug) task: Intended for charging electric vehicles by robot system automatically. In *International Multi-Conference on Systems, Signals and Devices*, 2015.

Niclas Bergman and Niclas Bergman. Recursive bayesian estimation: Navigation and tracking applications. thesis no 579. Technical report, Linköping University, Linköping Studies in Science and Technology. Doctoral dissertation, 1999.

Daniel Bernoulli. Exposition of a New Theory on the Measurement of Risk (1748). *Econometrica*, 22(1):23–36, Jan 1954. URL http://dx.doi.org/10.2307/1909829.

Aude Billard and Daniel Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2013.

Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Handbook of Robotics*, pages 1371–1394. 2008.

Christopher Bishop. *Pattern Recognition and Machine Learning*. 2006. URL http://research.microsoft.com/en-us/um/people/cmbishop/prml/.

Haitham Bou-Ammar, Holger Voos, and Wolfgang Ertel. Controller design for quadrotor uavs using reinforcement learning. In *International Conference on Control Applications*, pages 2130–2135, Sept 2010. URL http://dx.doi.org/10.1109/CCA.2010.5611206.

Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems (NIPS)*, volume 7, pages 369–376. MIT Press, 1995.

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Solving continuous pomdps: Value iteration with incremental learning of an efficient space representation. In *International Conference on Machine Learning (ICML)*, volume 28, pages 370–378. JMLR Workshop and Conference Proceedings, May 2013. URL http://jmlr.org/proceedings/papers/v28/brechtel13.pdf.

Alex Brooks and Stefan Williams. A monte carlo update for parametric pomdps. In *Robotics Research*, volume 66, pages 213–223. 2011. URL http://dx.doi.org/10.1007/978-3-642-14743-2_19.

Neil Burgess. Spatial memory: how egocentric and allocentric combine. *Trends in Cognitive Sciences*, 10(12):551–557, 2006. URL http://dx.doi.org/10.1016/j.tics.2006.10.005.

Lucian Busoniu, Damien Ernst, Bart de Schutter, and Robert. Babuska. Approximate reinforcement learning: An overview. In *Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8, April 2011. URL http://dx.doi.org/10.1109/ADPRL.2011.5967353.

Jesse Butterfield, Odest Jenkins, David Sobel, and Jonas Schwertfeger. Modeling aspects of Theory of Mind with Markov Random Fields. *Journal of Social Robotics*, 1(1):41–51, January 2009.

Sylvain Calinon, Florent D'halluin, Eric Sauser, Darwin Caldwell, and Aude Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics Automation Magazine*, 17(2):44–54, jun 2010. URL http://dx.doi.org/10.1109/MRA.2010.936947.

Henry Carrillo, Ian Reid, and José Castellanos. On the comparison of uncertainty criteria for active slam. In *International Conference on Robotics and Automation (ICRA)*, pages 2080–2087, May 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6224890.

Anthony Cassandra, Leslie Kaelbling, and James Kurien. Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 963–972 vol.2, Nov 1996.

Dong Chen and Georg von Wichert. An uncertainty-aware precision grasping process for objects with unknown dimensions. In *International Conference on Robotics and Automation (ICRA)*, pages 4312–4317, May 2015. URL http://dx.doi.org/10.1109/ICRA.2015.7139794.

Hongtai Cheng and Heping Chen. Online parameter optimization in robotic force controlled assembly processes. In *International Conference on Robotics and Automation (ICRA)*, pages 3465–3470, may 2014. URL http://dx.doi.org/10.1109/ICRA.2014.6907358.

Guillaume de Chambrier and Aude Billard. Learning search behaviour from humans. In *International Conference on Robotics and Biomimetics (ROBIO)*, pages 573–580, 2013. URL http://dx.doi.org/10.1109/ROBIO.2013.6739521.

Guillaume de Chambrier and Aude Billard. Learning search policies from humans in a partially observable context. *Robotics and Biomimetics*, 1(1):1–16, 2014. URL http://dx.doi.org/10.1186/s40638-014-0008-1.

Guillaume de Chambrier and Aude Billard. Fitted policy iteration for a pomdp peg-in-hole search task. *Robotics and Autonomous Systems*, 2016a.

Guillaume de Chambrier and Aude Billard. Non-parametric bayesian state space estimator for negative information. *Frontiers in Robotics and AI*, 2016b.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2011. URL http://dx.doi.org/10.1561/2300000021.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013. URL http://dx.doi.org/10.1561/2300000021.

Sandra Devin and Rachid Alami. An implemented theory of mind to improve human-robot shared plans execution. In *International Conference on Human Robot Interation (HRI)*, pages 319–326, 2016. URL http://dx.doi.org/10.1109/HRI.2016.7451768.

Yanzhu Du, David Hsu, Hanna Kurniawati, Wee Lee, Sylvie Ong, and Shao Png. A pomdp approach to robot motion planning under uncertainty. In *International Conference on Automated Planning and Scheduling, Workshop on Solving Real-World POMDP Problems*, 2010.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics Automation Magazine*, 13(2):99–110, June 2006. URL http://dx.doi.org/10.1109/MRA.2006.1638022.

Tom Erez and William Smart. A scalable method for solving high-dimensional continuous pomdps using local approximation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005a.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Jounral of Machine Learning Research*, 6:503–556, Dec 2005b.

William Fisher and Shahid Mujtaba. Hybrid position/force control: A correct formulation. *The International Journal of Robotics Research (IJRR)*, 11(4): 299–311, 1992. URL http://dx.doi.org/10.1177/027836499201100403.

Héctor González-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research (IJRR)*, 21(10-11):829–848, 2002. URL http://dx.doi.org/10.1177/027836402021010834.

Geoffrey Gordon. Stable function approximation in dynamic programming. In *International Conference on Machine Learning (ICML)*. Carnegie Mellon University, 1995. URL "http://www.cs.cmu.edu/~ggordon/ml95-stable-dp.ps.gz".

Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. URL http://dx.doi.org/10.1109/MITS.2010.939925.

Ivo Grondman, Lucian Busoniu, Gabriel Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, Nov 2012. URL http://dx.doi.org/10.1109/TSMCC.2012.2218595.

Vijaykumar Gullapalli, Andrew Barto, and Roderic Grupen. Learning admittance mappings for force-guided assembly. In *International Conference on Robotics and Automation (ICRA)*, pages 2633–2638, may 1994. URL http://dx.doi.org/10.1109/ROBOT.1994.351117.

Bradley Hamner, Sanjiv Singh, and Sebastian Scherer. Learning obstacle avoidance parameters from operator behavior. *Field Robotics*, 23(11/12):1037–1058, December 2006. URL http://dx.doi.org/10.1002/rob.20171.

Kris Hauser. *International Workshop on the Algorithmic Foundations of Robotics*, volume 9, chapter Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces, pages 193–209. 2011. URL http://dx.doi.org/10.1007/978-3-642-17452-0_12.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015. URL https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673.

Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *International Conference on Robotics and Automation (ICRA)*, pages 1814–1820, May 2008. URL http://dx.doi.org/10.1109/ROBOT.2008.4543471.

Paul Hebert, Thomas Howard, Nicolas Hudson, Jeremy Ma, and Joel Burdick. The next best touch for model-based localization. In *International Conference on Robotics and Automation (ICRA)*, pages 99–106, May 2013. URL http://dx.doi.org/10.1109/ICRA.2013.6630562.

Jan Hoffman, Michael Spranger, Daniel Gohring, and Matthias Jüngel. Making use of what you don't see: negative information in markov localization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2947–2952, Aug 2005. URL http://dx.doi.org/10.1109/IROS.2005.1545087.

Jan Hoffmann, Michael Spranger, Daniel Gohring, Matthias Jüngel, and Hans-Dieter Burkhard. Further studies on the use of negative information in mobile robot localization. In *International Conference on Robotics and Automation (ICRA)*, pages 62–67, May 2006. URL http://dx.doi.org/10.1109/ROBOT.2006.1641162.

Geoffrey Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav Sukhatme. Uncertainty-driven view planning for underwater inspection. In *International Conference on Robotics and Automation (ICRA)*, pages 4884–4891, May 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6224726.

Kaijen Hsiao, Leslie Kaelbling, and Tomás Lozano-Pérez. Task-driven tactile exploration. In *Robotics Science and Systems (RSS)*, Zaragoza, Spain, June 2010. URL http://dx.doi.org/10.15607/RSS.2010.VI.029.

Yifeng Huang and Kamal Gupta. Rrt-slam for motion planning with motion and map uncertainty for robot exploration. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1077–1082, Sept 2008. URL http://dx.doi.org/10.1109/IROS.2008.4651183.

Marco Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe Hanebeck. On entropy approximation for gaussian mixture random vectors. In *Multisensor Fusion and Integration*, pages 181–188, 2008. URL http://dx.doi.org/10.1109/mfi.2008.4648062.

Tina Iachini, Gennaro Ruggiero, and Francesco Ruotolo. Does blindness affect egocentric and allocentric frames of reference in small and large scale spaces? *Behavioural Brain Research*, 273(0):73–81, 2014. URL http://dx.doi.org/10.1016/j.bbr.2014.07.032.

Shervin Javdani, Matthew Klingensmith, Drew Bagnell, Nancy Pollard, and Siddhartha Srinivasa. Efficient touch based localization through submodularity. *International Conference on Intelligent Robots and Systems (IROS)*, pages 4013–4020, Nov 2013.

Matthew Johnson and et. al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015. URL http://dx.doi.org/10.1002/rob.21571.

Leslie Pack Kaelbling, Michael Littman, and Anthony Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, May 1998. URL http://dx.doi.org/10.1016/S0004-3702(98)00023-X.

Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, and Stefan Schaal. Learning force control policies for compliant manipulation. In *International Conference on Intelligent Robots and Systems (ICRA)*, pages 4639–4644, Sept 2011. URL http://dx.doi.org/10.1109/IROS.2011.6095096.

Michael Kasper, Gernot Fricke, Katja Steuernagel, and Ewald von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, 34(2):153–164, February 2001. URL http://www.sciencedirect.com/science/article/pii/S0921889000001196.

Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Transactions on Robotics and Automation (TRO)*, 12(4):566–580, Aug 1996. URL http://dx.doi.org/10.1109/70.508439.

Jin Kim, Michael Jordan, Shankar Sastry, and Andrew Ng. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 799–806. 2004. URL http://papers.nips.cc/paper/2455-autonomous-helicopter-flight-via-reinforcement-learning.pdf.

Jens Kober and Jan Peters. Learning motor primitives for robotics. In *International Conference on Robotics and Automation (ICRA)*, pages 2112–2118, May 2009. URL http://dx.doi.org/10.1109/ROBOT.2009.5152577.

Jens Kober, Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research (IJRR)*, July 2013.

Thomas Kollar and Nicholas Roy. Efficient optimization of information-theoretic exploration in slam. In *National Conference on Artificial Intelligence (AAAI)*, volume 3, pages 1369–1375, 2008. URL http://dl.acm.org/citation.cfm?id=1620270.1620287.

Seung kook Yun. Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion? In *International Conference on Robotics and Automation (ICRA)*, pages 1647–1652, May 2008. URL http://dx.doi.org/10.1109/ROBOT.2008.4543437.

Petar Kormushev, Sylvain Calinon, and Darwin Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3232–3237, October 2010a. URL http://dx.doi.org/10.1109/IROS.2010.5649089.

Petar Kormushev, Sylvain Calinon, Ryo Saegusa, and Giorgio Metta. Learning the skill of archery by a humanoid robot icub. In *International Conference*

on Humanoid Robots (Humanoids), pages 417–423, Dec 2010b. URL http://dx.doi.org/10.1109/ICHR.2010.5686841.

Klas Kronander. Control and learning of compliant manipulation skills, 2015.

Klas Kronander and Aude Billard. Online learning of varying stiffness through physical human-robot interaction. In *International Conference on Robotics and Automation (ICRA)*, pages 1842–1849, May 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6224877.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics Science and Systems (RSS)*, 2008.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

Pamela Banta Lavenexa, Valérie Boujonb, Angélique Ndarugendamwob, and Pierre Lavenexa. Human short-term spatial memory: Precision predicts capacity. *Cognitive Psychology*, 77:1–19, 2015.

Alan M. Leslie. *ToMM, ToBY, and Agency: Core architecture and domain specificity*. 1994.

Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems (RAS)*, 75, Part B: 352–364, 2016. URL http://dx.doi.org/10.1016/j.robot.2015.09.008.

Xin Li, William Cheung, and Jiming Liu. Improving POMDP Tractability via Belief Compression and Clustering. *Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(1):125–136, Feb 2010. URL http://dx.doi.org/10.1109/tsmcb.2009.2021573.

Georgios Lidoris. *State Estimation, Planning, and Behavior Selection Under Uncertainty for Autonomous Robotic Exploration in Dynamic Environments*. kassel university press GmbH, Kassel University, 2011.

Yong Lin, Xingjia Lu, and Fillia Makedon. Approximate planning in pomdps via MDP heuristic. In *International Joint Conference on Neural Networks (2014)*, pages 1304–1309, 2014. URL http://dx.doi.org/10.1109/IJCNN.2014.6889576.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning (ICML)*, 1995.

Arulampalam M. Sanjeev, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Transactions on Signal Processing*, 50(2):174–188, February 2002. URL http://dx.doi.org/10.1109/78.978374.

Jose Ramon Medina, Dominik Sieber, and Sandra Hirche. Risk-sensitive interaction control in uncertain manipulation tasks. In *ICRA*, pages 502–507, 2013.

Wim Meeussen, Melonee Wise, Stuart Glaser, Sachin Chitta, and et. al. Autonomous door opening and plugging in with a personal robot. In *International Conference on Robotics and Automation (ICRA)*, pages 729–736, May 2010. URL http://dx.doi.org/10.1109/ROBOT.2010.5509556.

George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information, 1956. URL http://cogprints.org/730/.

Volodymyr Mnih. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. URL http://dx.doi.org/10.1038/nature14236.

Michael Montemerlo and Sebastian Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1985–1991, Sept 2003. URL http://dx.doi.org/10.1109/ROBOT.2003.1241885.

Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Conference on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003. URL http://dl.acm.org/citation.cfm?id=1630659.1630824.

Bojan Nemec, Fares Abu-Dakka, Barry Ridge, and et. al. Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile. In *International Conference on Advanced Robotics (ICAR)*, pages 1–7, Nov 2013. URL http://dx.doi.org/10.1109/ICAR.2013.6766568.

Gerhard Neumann and Jan Peters. Fitted q-iteration by advantage weighted regression. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1177–1184. 2009a.

Gerhard Neumann and Jan Peters. Fitted q-iteration by advantage weighted regression. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1177–1184, Jun 2009b.

Andrew Ng and Michael Jordan. Pegasus: A policy search method for large mdps and pomdps. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, UAI'00, pages 406–415, 2000. URL http://dl.acm.org/citation.cfm?id=2073946.2073994.

D. Warner North. A tutorial introduction to decision theory. *Transactions on Systems Science and Cybernetics*, 4:200–210, 1968. URL http://dx.doi.org/10.1109/tssc.1968.300114.

A. Nowé, P. Vrancx, and Y-M. De Hauwere. *Reinforcement Learning: State-of-the-Art*, chapter Game Theory and Multi-agent Reinforcement Learning, pages 441–470. 2012. URL http://www.springer.com/engineering/computational+intelligence+and+complexity/book/978-3-642-27644-6.

Jose Nunez-Varela, B. Ravindran, and Jeremy Wyatt. Where do i look now? gaze allocation during visually guided manipulation. In *International Conference on Robotics and Automation (ICRA)*, pages 4444–4449, May 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6225226.

Dirk Ormoneit and Peter Glynn. Kernel-based reinforcement learning in average-cost problems. *Transactions on Automatic Control*, 47(10):1624–1636, Oct 2002. URL http://dx.doi.org/10.1109/TAC.2002.803530.

Hyeonjun Park, Ji-Hun Bae, Jae-Han Park, Moon-Hong Baeg, and Jaeheung Park. Intuitive peg-in-hole assembly strategy with a compliant manipulator. In *International Symposium on Robotics (ISR)*, pages 1–5, Oct 2013. URL http://dx.doi.org/10.1109/ISR.2013.6695699.

Achille Pasqualotto, Mary Jane Spiller, Ashok Jansari, and Michael Proulx. Visual experience facilitates allocentric spatial representation. *Behavioural Brain Research*, 236(0):175 – 179, 2013. URL http://dx.doi.org/10.1016/j.bbr.2012.08.042.

Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 365–371, Sept 2011. URL http://dx.doi.org/10.1109/IROS.2011.6095059.

Jan Peters and Stefan Schaal. Natural actor-critic. *European Symposium on Artificial Neural Networks*, 71(7-9):1180–1190, 2008a. URL http://dx.doi.org/10.1016/j.neucom.2007.11.026.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9): 1180–1190, mar 2008b. URL http://dx.doi.org/10.1016/j.neucom.2007.11.026.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, August 2003.

Christian Plagemann, Kristian Kersting, Patrick Pfaff, and Wolfram Burgard. Gaussian beam processes: A nonparametric bayesian measurement model for range finders. In *Robotics Science and Systems (RSS)*, June 2007. URL http://dx.doi.org/10.15607/RSS.2007.III.018.

Robert Platt, Russell Tedrake, Leslie Kaelbling, and Tomás Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics Science and Systems Conference (RSS)*, 2010. URL http://groups.csail.mit.edu/robotics-center/public_papers/Platt10.pdf.

Robert Platt, Leslie Kaelbling, Tomas Lozano-Perez, and Russ Tedrake. Non-gaussian belief space planning: Correctness and complexity. In *International Conference on Robotics and Automation (ICRA)*, pages 4711–4717, May 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6225223.

Josep Porta, Nikos Vlassis, Matthijs Spaan, and Pascal Poupart. Point-based value iteration for continuous pomdps. *Journal of machine learning research*, 7:2329–2367, 2006.

Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 8(11-12):1448–1465, December 2009.

Kerstin Preuschoff, Peter Mohr, and Ming Hsu. Decision making under uncertainty. *Frontiers in Neuroscience*, 7(218), 2013. URL http://dx.doi.org/10.3389/fnins.2013.00218.

Akshara Rai, Guillaume de Chambrier, and Aude Billard. Learning from failed demonstrations in unreliable systems. In *International Conference on Humanoid Robots (Humanoids)*, pages 410–416, 2013. URL http://dx.doi.org/10.1109/HUMANOIDS.2013.7030007.

Hilary Richardson, Chris Bake, Joshua Tenenbaum, and Rebecca Saxe. The development of joint belief-desire inferences. *Cognitive Science Societty*, 2012. URL http://dl.acm.org/citation.cfm?id=648205.749450.

Martin Riedmiller. *Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method*, pages 317–328. 2005. URL http://dx.doi.org/10.1007/11564096_32.

Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for pomdps. *Journal Artifcial Intelligence Research*, 32(1):663–704, Jul 2008. URL http://dl.acm.org/citation.cfm?id=1622673.1622690.

Nicholas Roy. Finding Approximate POMDP solutions Through Belief Compression. *Journal of Artificial Intelligence Research*, 23, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.6180.

Nicholas Roy and Geoffrey Gordon. Exponential family pca for belief compression in pomdps. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 1667–1674. MIT Press, 2003. URL http://papers.nips.cc/paper/2319-exponential-family-pca-for-belief-compression-in-pomdps.pdf.

Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems (NIPS)*, volume 12, pages 1043–1049, 1999.

Nicholas Roy, Wolfram Burgard, Dieter Fox, and Thrun Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *International Conference on Robotics and Automation (ICRA)*, pages 35–40, May 1999.

Brian Scassellati. Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1):13–24, January 2002. URL http://dx.doi.org/10.1023/A:1013298507114.

Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In *11th International Symposium on Robotics Research (ISRR)*, 2004. URL http://dx.doi.org/10.1007/11008941_60.

Ross D. Shachter. Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 480–487, 1998. URL http://dl.acm.org/citation.cfm?id=2074094.2074151.

Chhatpar Siddharth and Michael Branicky. Search strategies for peg-in-hole assemblies with position uncertainty. In *International Conference on Intelligent Robots and Systems (ICRA)*, volume 3, pages 1465–1470, 2001. URL http://dx.doi.org/10.1109/IROS.2001.977187.

David Silver, J. Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *IJRR*, 29(12):1565–1592, October 2010. URL http://dx.doi.org/10.1177/0278364910369715.

Richard Smallwood and Edward Sondik. The optimal control of partially observable markov processes over a finite horizon. *Oper. Res.*, 21(5):1071–1088, Oct 1973. URL http://dx.doi.org/10.1287/opre.21.5.1071.

Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, Arlington, Virginia, United States, 2004. AUAI Press.

Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005. URL http://arxiv.org/abs/1207.1412.

Beate Sodian and Susanne Kristen. Theory of mind. In *Towards a Theory of Thinking*, pages 189–201. 2010. URL http://dx.doi.org/10.1007/978-3-642-03129-8_13.

Matthijs Spaan and Nikos Vlassis. Planning with continuous actions in partially observable environments. In *International Conference on Robotics and Automation (ICRA)*, pages 3469–3474, 2005.

Cyrill Stachniss, Grisetti Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics Science and Systems (RSS)*, 2005.

Brian Stankiewicz, Gordon Legge, Stephen Mansfield, and Erik Schlicht. Lost in virtual space: Studies in human and ideal spatial navigation. *Journal of Experimental Psychology: Human Perception and Performance.(under review)*, 32(3):688–704, 2006.

Katie Steele and H. Orri Stefansson. Decision theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2015 edition, 2015.

Freek Stulp, Evangelos Theodorou, Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning motion primitive goals for robust manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 325–331, Sept 2011. URL http://dx.doi.org/10.1109/IROS.2011.6094877.

Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Transactions on Robotics (TRO)*, 28(6):1360–1370, Dec 2012. URL http://dx.doi.org/10.1109/TRO.2012.2210294.

Wen Sun and Ron Alterovitz. Motion planning under uncertainty for medical needle steering using optimization in belief space. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1775–1781, Sept 2014. URL http://dx.doi.org/10.1109/IROS.2014.6942795.

Hsi Guang Sung. *Gaussian Mixture Regression and Classification*. PhD thesis, Rice University, 2004.

Richard Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 1057–1063. MIT Press, 2000.

Richard S. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2010. URL http://dx.doi.org/10.2200/S00268ED1V01Y201005AIM009.

Sebastian Thrun. Monte carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1064–1070, 2000.

Sebastian Thrun. Particle filters in robotics. In *Annual Conference on Uncertainty in AI (UAI*, volume 17, 2002.

Sebastian Thrun and Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. In *National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 944–950, 1996. URL http://dl.acm.org/citation.cfm?id=1864519.1864527.

Sebastian Thrun and John J. Leonard. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 871–889. 2008. URL http://dx.doi.org/10.1007/978-3-540-30301-5_38.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

Rafael Valencia, Jaime Valls Miro, Gamini Dissanayake, and Juan Andrade-Cetto. Active pose slam. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1885–1891, Oct 2012. URL http://dx.doi.org/10.1109/IROS.2012.6385637.

Joan Vallve and Juan Andrade-Cetto. Dense entropy decrease estimation for mobile robot exploration. In *International Conference on Robotics and Automation (ICRA)*, pages 6083–6089, 2014. URL http://dx.doi.org/10.1109/ICRA.2014.6907755.

Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *Int. J. Rob. Res.*, 30(7):895–913, Jun 2011. URL http://dx.doi.org/10.1177/0278364911406562.

Jur van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *International Journal of Robotics Research (IJRR)*, 31(11):1263–1278, 2012. URL http://dx.doi.org/10.1177/0278364912456319.

Tiago Veiga, Matthijs Spaan, and Pedro Lima. Point-based POMDP solving with factored value function approximation. In *Conference on Artificial Intelligence (AAAI)*, volume 28, pages 2512–2518, 2014.

Manuela M. Veloso, editor. *International Joint Conference Artificial Intelligence (IJCAI)*, Jan 2007.

Ngo Vien and Marc Toussaint. Pomdp manipulation via trajectory optimization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 242–249, Sept 2015a.

Ngo Vien and Marc Toussaint. Pomdp manipulation via trajectory optimization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 242–249, Sept 2015b. URL http://dx.doi.org/10.1109/IROS.2015.7353381.

Sethu Vijayakumar, Tomohiro Shibata, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Autonomous Robot*, 2003.

John von Neumann and Oskar Morgenstern. *The theory of games and economic behavior*. Princeton, 3 edition, 1990.

Frances Wang. Spatial updating. *Scholarpedia*, 2(10):3839, 2007.

Jiexin Wang, Eiji Uchibe, and Kenji Doya. Em-based policy hyper parameter exploration: application to standing and balancing of a two-wheeled smartphone robot. *Artificial Life and Robotics*, 21(1):125–131, 2016. URL http://dx.doi.org/10.1007/s10015-015-0260-7.

Ranxiao Frances Wang and Elizabeth Spelke. Updating egocentric representations in human navigation. *Cognition*, 77(12):215 – 250, 2000. URL http://www.wjh.harvard.edu/~lds/pdfs/wang2000.pdf.

Marco Wiering and Martijn van Otterio. *Reinforcement Learning State-of-the-Art*. Springer-Verlag Berlin Heidelberg, 2012.

Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. URL http://dx.doi.org/10.1007/BF00992696.

David Winter. *Biomechanics and motor control of human movement*. 2009.

Thomas Wolbers and Mary Hegarty. What determines our navigational abilities? *Trends in Cognitive Sciences*, 14(3):138 – 146, 2010. URL http://dx.doi.org/10.1016/j.tics.2010.01.001.

Thomas Wolbers, Mary Hegarty, Christian Buchel, and Jack Loomis. Spatial updating: how the brain keeps track of changing object locations during observer motion. *Nature Neuroscience*, 1124(1), 2008. URL http://dx.doi.org/10.1038/nn.2189.

Yang Yang, Linglong Lin, Y. T. Song, Bojan Nemec, Ales Ude, Anders Glent Buch, Norbert Kruger, and Thiusius Rajeeth Savarimuthu. *Fast programming of peg-in-hole actions by human demonstration*, pages 990–995. 2014. URL http://dx.doi.org/10.1109/ICMC.2014.7231702.

Tingting Zhao, Gang Niu, Ning Xie, Jucheng Yang, and Masashi Sugiyama. Regularized policy gradients: Direct variance reduction in policy gradient estimation. In *Proceedings of the Fourth Asian Conference on Machine Learning (ACML)*, volume 45 of *JMLR Workshop and Conference Proceedings*, pages 333–348, Nov 2015.

Claudio Zito, Marek Kopicki, Rustam Stolkin, Christoph Borst, Florian Schmidt, Maximo Roa, and Jeremy Wyatt. Sequential trajectory re-planning with tactile information gain for dexterous grasping under object-pose uncertainty. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4013–4020, Nov 2013. URL http://dx.doi.org/10.1109/IROS.2013.6696930.

Guillaume de Chambrier

Avenue de la Gare 3 – 1020 Renens – Switzerland • Born 29.06.1987
079 822 76 35 • chambrierg@gmail.com
http://lasa.epfl.ch/people/member.php?SCIPER=213946.com • Swiss and British

Profile

With several independent projects, I am an expert in developing and applying machine learning techniques to robot systems and I poses meticulous strong analytical skills.

Education

PhD in Manufacturing Systems & Robotics — **École polytechnique fédérale de Lausanne, Switzerland**
- Thesis: Learning Search Strategies from Human Demonstrations — *2012 - 31.08.2016*
  supervisor: Prof. Aude Billard

First class Master of Informatics with Honours Informatics — **University of Edinburgh, UK**
- Thesis: Building and Controlling a Hexapod Robot — *2006 - 2011*
  supervisor: Dr. Michael Herrmann

Erasmus Exchange, Bachelor — **Universität des Saarlandes, Germany**
2008 - 2009

Experience

Teaching Assistant — **École polytechnique fédérale de Lausanne**
- Course: Applied Machine Learning (MSc) — *2013-2016*
- Course: Advanced Machine Learning (MSc)

European Project — **École polytechnique fédérale de Lausanne**
- Flexible Skill and Intuitive Robot Tasking — *2012-2013*

Supervision — **École polytechnique fédérale de Lausanne**
- Akshara Rai (Msc student) — *2013*

Technical Skills

Programming: C/C++, Python, Java, MATLAB

Expertise: Robotics, Reinforcement Learning, Non-parametric Bayesian inference, Machine learning & Computer Vision

Languages

French, English (first language)

Awards and Certification

2010: Google Prize: Best Phase 1 Project in Master of Informatics Programme

Publication

A. Rai, G. de Chambrier, A. Billard: Learning from Failed Demonstrations in Unreliable Systems. International Conference on Humanoid Robots, Oct. 2014

G. de Chambrier, A. Billard: Learning search behaviour from humans. International Conference on Robotics and Biomimetics, Dec. 2013

G. de Chambrier, A. Billard: Learning search policies from humans in a partially observable context. Journal of Robotics and Biomimetics, 2014

G. de Chambrier, A. Billard: Fitted Policy Iteration for a POMDP Peg-In-Hole search task. Journal of Robotics and Autonomous Systems, 2016

G. de Chambrier, A. Billard: Non-parametric Bayesian State Space Estimator for Negative Information. Frontiers in Robotics and AI, 2016 *(under review)*