Learning Forward-Models for Robot Manipulation

Michael Jacob Mathew

University of Birmingham

The School of Computer Science

This dissertation is submitted for the degree of

Doctor of Philosophy

Birmingham **April, 2020**

to
Carlin
Acha
Amma
Deepa

Acknowledgements

My PhD course has been a life-changing experience. It would have been impossible without the help, support and guidance from my peers, friends and family. First and foremost, I am greatly indebted to Prof. Jeremy L Wyatt, who made this journey possible. His strong recommendation helped me to win the prestigious Commonwealth Scholarship (INCS-2015-230) and enabled me to pursue my PhD at the IRLab, University of Birmingham. Our lively discussions, whiteboard exercises and his intriguing questions about my work, all guided me through this challenging journey. I cannot thank Prof. Mohan Sridharan enough for all the support and guidance in the most crucial part of my PhD. His talent at pointing out mistakes and shortcomings in a positive manner was helpful during my most stressful moments. I have not met many professors like Jeremy and Mohan who take such immense pains to support their students. I feel privileged to have met and worked under them. I gratefully acknowledge the Commonwealth Scholarship Commission (CSC) for considering my application and providing me with adequate financial support to undertake this course.

The in-depth knowledge of Dr Morteza Azad on robot dynamics and control was an indispensable resource in the past three years. I much appreciate his thought-provoking questions about my ideas which always triggered a new perspective and understanding. Prof. Ales Leonardis is another person I would like to thank for being on my thesis committee for all four years of my PhD. I cannot thank Ermano Arruda enough for being a patient mentor who lent support and technical help without any hesitation. We built a software stack and collaborated for a research publication which formed a part of my thesis contribution. In March 2019, his timely insight also helped me figure out a bug in the robot SDK that hindered our progress. Fixing that bug restored my hope of completing this journey. I am immensely happy to have collaborated with him. I met Saif Sidhik during my second year and little did I know how he would provide immense help in setting up my experiments and generous support during all my failures. I am obliged to Bertram Dandy from the Computer Science workshop for his assistance for the experimental setups in this thesis. I am also happy to have met wonderful people like Diar, Aki, Maxime and Laura during this journey.

Next is my lovely wife, Carlin. She provided the best moral and emotional support in the final lap of the journey. I want to express my overwhelming gratitude for her contagious

positive energy that helped me surmount this hardship. The very few days I didn't live in the lab I spent with my dear wife and close friends. Saif, Cyriac, Sam, Jithin are all valuable relationships I made during the past few years. We hung out at house parties, outside dinners, movie nights and weekend trips. I think the best memory from my PhD journey will be the precious and priceless moments I got to be with them. My parents, Acha and Amma, sister Deepa, have always been there to support me. I believe it is their prudence, prayers and goodwill that helped me in achieving everything I have today. Looking back on the 23rd of September 2015, I realise I started this journey with a massive bias of *Dunning Krugger effect*. At the end of this long journey, however, I feel a considerable reduction in this bias and have acquired the ability to look at a problem with more clarity. Last but not least, I want to thank the Almighty for making all the odds against this journey manageable. So many things could have gone wrong, but I survived this far; I feel humbled and blessed!

Abstract

Robots with the capability to dexterously manipulate objects have the potential to revolutionise automation. Interestingly, human beings have the ability to perform complex object manipulations. They perfect their skills to manipulate objects through repeated trials. There is extensive human motor control literature which provides evidence that the repetition of a task creates forward-models of that task in the brain. These forward-models are used to predict future states of the task, anticipate necessary control actions and adapt impedance quickly to match task requirements. Evidence from motor control and some promising results in the robot research on manipulation clearly shows the need for forward-models for manipulation.

This study was started with the premise that a robot needs forward-models to perform dexterous manipulation. Initially planning a sequence of actions using a forward-model was identified as the most crucial problem in manipulation. Push manipulation planning using forward-models was the first step in this direction. However, unlike most methods in the robotic push manipulation literature, the approach was to incorporate the uncertainty of the forward-model in formulating the plan for push planning. Incorporating uncertainty helps the robot to perform risk-aware actions and stay close to the known areas of the state space while manipulating the object. The forward-models of object dynamics were learned offline, and robot pushes were fixed-duration position-controlled actions. The experiments in simulation and real robots were successful and helped in creating several other insights for better manipulation. Two of these insights were the need to have the capability to learn the feed-forward model online and the importance of having a state-dependent stiffness controller.

The first part of the thesis presents a planner that makes use of an uncertain, learned, forward (dynamical) model to plan push manipulation. The forward-model of the system is learned by poking the object in random directions. The learned model is then utilised by a model predictive path integral controller to push the box to the required goal pose. By using path-integral control, the proposed planner can find efficient paths by sampling. The planner is agnostic to the forward-model used and produces successful results using a physics simulator, an Ensemble of Mixture Density Networks (Ensemble-MDN) or a Gaussian

Process (GP). Both ensemble-MDN and a GP can encode uncertainty not only in the push outcome but in the model itself. The work compares planning using each of these learned models to planning with a physics simulator. Two versions of the planner are implemented. The first version makes uncertainty averse push actions by minimising uncertainty cost and goal costs together. Using multiple costs makes it difficult for the optimiser to find optimal push actions. Hence the second version solves the problem in two stages. The first stage creates an uncertainty averse path, and the second stage finds push actions to follow the path found.

The second part of the thesis describes a framework which can learn forward-models online and can perform state-dependent stiffness adaptation using these forward-models. The idea of the framework is again motivated by the human control literature. During the initial trials of a novel manipulation task, humans tend to keep their arms stiff to reduce the effects of any unforeseen disturbances on the ability to perform the task accurately. After a few repetitions, humans adapt the stiffness of their arms without any significant reduction in task performance. Research in human motor control strongly indicates that humans learn and continuously revise internal models of manipulation tasks to support such adaptive behaviour. Drawing inspiration from these findings, the proposed framework supports online learning of a time-independent forward-model of a manipulation task from a small number of examples. The proposed framework consists of two parts. The first part can create forward-models of a task through online learning. Later, the measured inaccuracies in the predictions of this model are used to dynamically update the forward-model and modify the impedance parameters of a feedback controller during task execution. Furthermore, the framework includes a hybrid force-motion controller that enables the robot to be compliant in particular directions (if required) while adapting the impedance in other directions. These capabilities are illustrated and evaluated on continuous contact tasks such as polishing a board, pulling a non-linear spring and stirring porridge.

Table of contents

Chapter 1

Introduction

Manipulation, beyond the topic of grasping, is one of the most difficult and open areas of robotics research. Manipulation can be defined as the intentional modification of an object's configuration using specific action sequences. Some manipulation actions, for example grasping, have been widely studied in robotics. Beyond grasping, operations such as pushing, flipping, sliding, polishing and stirring still pose unsolved problems. The main challenge is the inability of existing control strategies to effectively facilitate dexterity to robot arms. The lack of human-like dexterity still confines robots to particular tasks. This is evident from the automotive industry where each task (e.g. painting, welding) requires a different robot. This thesis focuses on the dexterity aspect in robot arms that enables them to plan and anticipate compliant contact actions in uncertain environments.

Dexterity of human hands in manipulating objects play a pivotal role in our daily lives. Humans skillfully control force and position of their hands or tools to manipulate objects even in uncertain environments. They can learn and adapt skills online and be compliant to disturbing forces. However, the industry has generally favoured the use of stiff position-controlled robots over force-controlled robots. The affinity for stiff robots is mostly due to three reasons: (1) for fast, precise and repeatable motion in structured environments (2) most applications that require robot manipulators are deterministic and have access to an accurate description of the task and environment (3) lack of need for collaborative work with humans. Since frequent human interactions with these robots are minimal, it is possible to confine them to their work cells by building cages.

Industrial manipulators in structured environments function effortlessly using extensive knowledge of its working environment and assuming all the scenario that it may encounter is within its knowledge domain. The extensive domain knowledge helps to pre-program each robot's behaviour. Further, they can keep repeating this behaviour till a task or environment change occurs. However, it becomes difficult to program the robot when used in unknown or

partially known environments. Hence, it becomes essential to account for the environmental uncertainties during planning. For example, precise spatial knowledge of the tools, workpiece and environment helps a robot in an automotive industry to assemble cars. A slight inaccuracy in the position of the tools or the environment results in erroneous task execution. Subsequently the robot has to undergo re-calibration to continue the task.

Currently, there is an increase in demand for robots to work along with humans in unstructured environments without enclosures (e.g. furniture assembly works). The recent development of direct-torque control robots is a promising step towards making these collaborative work environments possible. However, analytically modelling the complex non-linear interaction of the robot with humans or its environment is cumbersome or impractical. Effective use of robots in unstructured environments require solving several challenges which hinder the use of existing frameworks. In particular, there are three difficulties: (i) planning manipulations require a model which can predict the effect of actions (ii) even with such a model there is uncertainty -both *aleatoric*[1] and *epistemic*[2] - in the outcome and hence this should be captured to allow optimal planning/control and finally (iii) there is a requirement to have some form of compliance behaviour to deal with changing external forces that act on the robot and for the manipulated object. However, there are other challenges to solve these problems such as (i) physics models are inaccurate and do not capture uncertainty (ii) planning methods proposed in the literature typically do not have the capability to account for the uncertainty in the model and (iii) when interacting with the environment, simple approaches to compliance give poor positional control and no explicit force control.

In this thesis we develop the control and planning framework that addresses all three problems. The core contributions show :

1. Using forward-models and their predictive uncertainty to plan push manipulations. In particular, how to learn and plan with kinematic models of manipulation effects that contain both aleatoric and epistemic uncertainty. We combine learned-forward models with stochastic planners to achieve uncertainty averse push manipulation. Our approach is significantly different from the existing literature (Section 2.2). Chapter 3 provides further particulars about the contribution and was published in Arruda et al. [9].

2. Extending this approach to predict the effects on felt forces in continuous interaction tasks by learning simple forward-models online. Prediction of forces in such tasks can help in anticipatory control of the interacting robot manipulator. The incorrectness

[1] random variability present in the system

[2] uncertainty that comes due to insufficient data/sensing capabilities

of the force prediction is used to improve the impedance control by making it a task-space, time-independent, state-dependent stiffness controller. This is unique to our framework and a departure from the existing methods (Section 2.3). The specifics of the framework proposed is covered in Chapter 4 and was published in Mathew et al. [107].

3. Later we describe on how to plan and control online, using this new control framework in a variety of continuous interaction tasks defined in static and dynamic environments. Details of this contribution are discussed in Chapter 5. Earlier version of the experiments are covered in Mathew et al. [108] and further improved in our work in Mathew et al. [107].

1.1 Approach

This section gives an outline of the different proposed approaches in this thesis. We start with giving an outline (Section 1.1.1) on how to plan push actions to move an object avoiding highly uncertain regions. Chapter 3 details this approach which forms our first contribution. The planning approach described in this chapter generates sequential and discrete push actions to move an object to the specified target pose. After the success with push planning which involved discrete interactions with the object, we moved to tasks that involved continuous interaction with the environment. A framework to solve such continuous interaction tasks using variable impedance constitutes our second and third contributions. Section 1.1.2 provides an outline of our proposed variable impedance approach.

1.1.1 Uncertainty averse push planning

Consider an example similar to that discussed in Doshi-Velez [41]. Imagine that you are in a new city with an incomplete map of the place and you only know a few routes in the city. Suppose, you want to travel from place A to B within the city and the only route you know is via place C. Although it is an indirect and long route, it helps you avoid getting lost. If you are to take an unknown route, using the incomplete map, the certainty of you reaching the goal decreases. Hence it is better to account your uncertainty about various parts of the city while route planning. This accounting will help you to avoid high risk areas and move through the known parts of the city. Such route planning allows to trade-off risk for time to reach the destination. Similarly, it would be better to stick to the known parts of the city in order to avoid the risk of getting lost while travelling to a new destination. But sticking to

known regions within the city should not result in taking excess time to reach the destination. That means, there is an optimal trade-off between risk and time taken to reach the destination. So the question is how to come up with a optimal route or a *trajectory*[3] that stays within your known regions of the map.

A simple way to find a safe path is by using trajectory samples; that is by picking the best path out of a set of possible trajectories from start to finish. In this case, the best trajectory is the one that takes the minimum time to reach the destination while still being in the certain regions. However, searching for a single best trajectory is exhaustive. A better and intuitive way is to start with a potential trajectory from start to finish and then incrementally modifying it in such way to perfectly balance the trade-off. For such incremental updates on the starting trajectory, we will require a *cost function*. A cost-function is a mathematical equation which can quantify the merit of a trajectory. These merits serve as marks and enables ranking of trajectories thereby helping to select the best one. Further, we will require a *forward-model* if we want to find a sequence of inputs (e.g. efforts by leg muscles in our example) that will generate the chosen trajectory. A forward-model is the model of the system (here our motor control system) which can generate a trajectory using a sequence of inputs. We need to follow slightly modified set of steps to find out the sequence of inputs that results in the best trajectory. Now, we can start with a set of random inputs. We can feed these inputs to the forward-model to generate the resulting trajectory. The trajectory generated can be quantified for its merit using the cost-function. If we pick a set of random inputs and simulate each of them using the forward-model, we can get a set of trajectories and their corresponding merits. These trajectory merit values enables us to find the gradient of the cost-function. This gradient can be used to update the input sequence to generate trajectory that has least cost. (details in Chapter 3). There are different types of forward-models. Some of them can only simulate the input sequence in a deterministic fashion. However, there will be many cases where the system model may be unsure about a simulated outcome under an input. Such cases can only be provided with a forward-model which can also quantify its predictive uncertainty.

The forward-model used for our first contribution (Chapter 3) is learned from data and can estimate uncertainties of its predictions. The data is collected by observing the behaviour of the system under different actions. Since the data collected is not exhaustive, the trained forward-model is inexperienced of the complete state-space. Hence, the predictions made in less experienced areas will be uncertain or error prone. We find each action taken by the system (e.g. robot) using its forward-model predictions and therefore the predictive uncertainty contributes significantly towards the effectiveness of the action. For example,

[3] A time indexed sequence of poses followed by the system -here human

taking a particular action in some areas of the state space can transition the system to more uncertain regions. Occurrence of such events will result in choosing actions which are more or less random (since the choice of action depends on the prediction). Incorrect choice of actions in uncertain areas of the state space may have unwanted consequences. Hence, it will be highly beneficial to incorporate predictive uncertainty into planning. Chapter 3 describes a technique to incorporate the predictive uncertainty to trajectory planning. The approach uses the path integral based approach (Section 3.2.2) developed by Kappen [72] in the stochastic optimal control (Section 2.5.3) framework. We propose two algorithms in Chapter 3 to solve the problem of finding optimal actions to follow a less risky trajectory. First algorithm finds optimal actions at each state of the robot using a cost function which is a linear combination of *task cost* and *uncertainty cost*. While on a trajectory, the system passes through different task states. Task cost is a mathematical equation which penalises each system state for some task specific goals. Uncertainty cost is a mathematical equation which penalises uncertainty associated with each system state. The second algorithm finds optimal actions in two steps by decoupling task cost and uncertainty cost. Initially the uncertainty cost alone is used to find a less risky trajectory and subsequently the task cost alone is used to find optimal actions to follow this trajectory.

1.1.2 State dependent impedance control

Consider the task of stirring porridge with a spatula. As the task advances, the porridge starts to thicken, increasing the force felt at the palm. Correspondingly, to keep stirring, we often increase our arm stiffness. Similar such examples are inserting a bulb to a holder, turning of key in a lock or inserting a plug. We vary the stiffness of the arm to manipulate the bulb or the key to achieve the task goal. Performing these tasks with a constant stiffness increases their level of difficulty. Also, the choice of stiffness applied at each *state of the task* has a tremendous influence on the overall performance. Several such examples from our daily lives show that varying stiffness is an important aspect of human dexterity. Studies in psycho-physics (Franklin et al. [52], Burdet et al. [21]) strongly suggest that we learn to vary stiffness while training on a novel manipulation task. However, what seems easy and intuitive for humans in interactive tasks is difficult for robots. It is well understood that to achieve human-like performance, the robots needs to acquire the ability to vary their stiffness while performing a task.

Varying stiffness in robots is achieved by implementing controllers in the *impedance* [4] framework. Control of impedance of a mechanism regulates the motion due to the force felt as

[4]Mechanical impedance is defined as the ratio between force and motion.

a result of environment interactions (Hogan [62]). Increase in impedance of the manipulator allows in better rejection of perturbing forces, but makes it stiff. Correspondingly, decrease in manipulator impedance parameters makes it compliant to external forces. For a manipulator having task-redundant degrees of freedom, some of these external forces can be compensated by moving the arm in its null-space. For other forces which cannot be compensated by such null-space adjustments are taken care by trading off the arm's motion accuracy.

Some tasks that require control of force and motion along the same direction involve application of varying forces by adjusting its impedance parameters. However, adjustment of impedance parameters are not arbitrary and totally depends on the current task configuration, reaction force felt and intended state of the task. In this thesis, the impedance control is developed and deployed in the task-space of the robot. Task-space controllers (Cartesian-space controllers - Section 2.5.1) are intuitive and have task-specific parameters. This allows the controller to be independent of the type of robot manipulator used. These controllers are typically designed as a *mass-spring-damper* system. The goal of the controller is to make the robot behave like a spring attached between the end-effector and the target *way-point*. The spring behaviour of the end-effector is defined through three parameters; (1) *inertia tensor*[5] (2) *stiffness matrix*[6] and (3) *damping matrix* [7]. Variable impedance behaviour is achieved by modifying the inertia, damping and stiffness parameters. However, modification of these parameters is not trivial and is dependent on task and robot state. Shaping *inertia* of the robot to behave like a mass-spring-damper system is generally tricky (Wimbock et al. [166]) and requires accurate measurement of the external forces acting on the robot. Measurement of forces at every acting point is impractical and hence it is difficult to modulate the inertia tensor as it leads to unreliable impedance behaviour. For this reason, the desired impedance behaviour is often limited to designing *stiffness* and *damping* parameters of the controller, while keeping the end-effector inertia unchanged. This modified problem is called as the compliance control problem (Wimbock et al. [166]).

1.2 Thesis structure

This thesis is organised as follows. Chapter 2 illustrates essential background knowledge required to follow the thesis along with an extensive survey on planning of push manipulation

[5]The inertia (moment of inertia) is the tensor parameter which determines the torque required for a specified angular acceleration about an axis. This parameter is also called as angular mass or rotational inertia and is similar to how mass determines the required force for a specified acceleration. (Marion [102]) In the case of a robot manipulator, it is the effective inertia of the entire arm reflected at its end point.

[6]Stiffness can be defined as the parameter which determines the deviation of the robot end-effector from a waypoint under an external force.

[7]Damping can be defined as the parameter which determines oscillatory response of the robot end-effector due to its acceleration.

(Section 2.2), learning forward-models (Section 2.1) and variable impedance controllers (Section 2.3). Details of the two planners proposed in Contribution 1 are described in Chapter 3 along with experiments designed to validate them. Chapter 4 provides details on the learning framework developed for anticipatory control and variable impedance control for continuous interactions tasks by learning its forward-models (Contributions 2 and 3). The experiments conducted to validate variable stiffness framework are detailed in Chapter 5 (Contribution 4). Finally, Chapter 6 gives the conclusion of this research, the limitations of the proposed approaches and promising research directions to fix them. Appendix A provides a brief derivation of the task space controller formulation and Appendix B gives brief information about the hardware used in our experiments and the software stack developed as a part of this research. Finally, Appendix C discusses the alternate attempted approaches in the past years for solving the problems listed in this thesis.

1.3 Publications

1. Arruda, Ermano, Michael J. Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L. Wyatt. "Uncertainty averse pushing with model predictive path integral control." In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pp. 497-502. IEEE, 2017.

2. Michael J. Mathew, Saif Sidhik, Mohan Sridharan, Akinobu Hayashi, Morteza Azad, and Jeremy L. Wyatt. "Online Learning of Feed-Forward Models for Task-Space Variable Impedance Control" In 2019 IEEE-RAS 19th International Conference on Humanoid Robotics (Humanoids), pp. 597-602. IEEE, 2019. An earlier version of this work appeared in Mathew et al. [108].

The first publication is concerned with the initial contribution of this thesis which is an equally contributed work between myself and Arruda E. This work involved formulation of a forward-model and using it for push planning. The work in with this thesis is a stochastic planner which uses the forward-model proposed by Arruda E to generate optimal push actions to move an object towards a given goal pose. Chapter 3 covers more details of the planner proposed in this publication. The second publication involves second and third contributions of this thesis. This publication covers a state dependent variable impedance controller framework motivated by human motor control studies. Further details of the framework can read in Chapter 4. The framework is thoroughly tested on different experiments which are described in detail in Chapter 5.

Chapter 2

Related work and background

This chapter reviews the relevant literature and technical preliminaries required to understand the two frameworks introduced in Chapter 3 and Chapter 4. The framework described in Chapter 3 illustrates an approach to use the uncertainty in forward-models for planning push manipulation. Section 2.1 presents literature on usage of forward-models in robotics followed by Section 2.2 reviews different strategies used for push manipulation.

The push manipulation experiments described in Chapter 3 deals with discrete contact tasks. The success of our approach motivated the development of a framework for continuous interaction tasks. The framework developed can learn the forward-model of a continuous interaction task online and make use of its predictive uncertainty to modulate robot impedance parameters. Review on related approaches which learn variable impedance are described in Section 2.3 and on different control strategies are in Section 2.4. Finally, Section 2.5 gives essential background knowledge required to follow different methods and techniques used in this thesis.

2.1 Learning of forward-models from data

Robots are meant to interact with the environment through discrete (tasks like assembly) or continuous contacts (tasks like polishing, deburring etc). Such interactions require the knowledge of environment behaviour to select or modify appropriate action. However, it is impossible to model all properties and dynamics of an environment. One way to tackle this problem is by enabling the robot to learn from its own experience; that is, the robot observes (measures) the changes to the environment through its interactions and models them by using suitable machine learning techniques (supervised or reinforcement learning etc). Access to such a model (called a forward-model) helps the robot to simulate the effects of its interaction in advance. In other words, a forward-model enables a robot to predict the

next potential state based on its current state and action. This ability to learn forward-models enables robots to be more intelligent, adaptable and autonomous (Dearden et al. [33]). The work in Beetz et al. [12] shows the importance of learning models from observation to meet the demands of novel situations which cannot be met by stereotypical solutions.

Studies in psychophysics like Miall and Wolpert [114], Wolpert et al. [168], Haruno et al. [61], Flanagan et al. [50] and Mehta and Schaal [113] provides firm evidence for the existence of forward-models in human motor control. It has been hypothesised that forward and inverse models exist in the human brain (Wolpert et al. [169]) and was confirmed by the motor simulation created by Dindo et al. [39] using a Bayesian framework. A computational framework which uses this idea is the *hierarchical attentive multiple models* for modelling motor execution and recognition (Demiris and Khadhouri [36]). This idea was later used in robot experiments for imitation learning by (Demiris and Meltzoff [37]). Interesting experiments on human visuomotor control by Mehta and Schaal [113] suggests that forward-model based control is often easier to learn. This ease arises since the forward-models can be acquired through supervised or self-supervised learning.

Using forward-models are popular in several domains of robotics. There are several works in the literature which uses forward-models for online sensor calibration (Maye et al. [109]), learning occupancy grid maps (Thrun [158]) and prediction of intent in human-robot collaborative works (Demiris [35]). The work in Dearden et al. [33] proposes a Bayesian framework to learn the forward-model of a single DoF robot action. The work in Stulp and Beetz [152] uses *action models* which predict the effects of an action and show how such forward-models lead to more efficient, robust and effective behaviour.

In robot manipulation, Tian et al. [159] (though limited to a single finger) uses a forward-model created from high bandwidth tactile sensors to augment their measurements for better manipulation. The work in Boots et al. [16] shows an improvement to manipulation abilities when the system can anticipate future observation from a learnt model. Shon et al. [146] uses a Bayesian framework to create forward-models from the demonstrated samples to generate skills for a robot on real-time basis. Works like Stüber et al. [151] extensively describes forward-models used in push manipulation. Interested readers are further directed to the work by Nguyen-Tuong and Peters [121] on learning models for control in robots.

In general, forward-models are used to predict the behaviour of the robot (Kronander and Billard [81], Huang et al. [65]) and/or the objects (Fan et al. [46]) being effected while performing the task. The main challenge in the development of such a forward-model is the selection of relevant state features of the task. Effective learning and performance of a forward-model massively depends on the choice of the feature vector. Forward-models are of

three types (i) analytic model - derived using principles of physics (ii) learned model - from data (iii) hybrid model - combination of learned and analytic models

One approach for using predictive models is to build a analytic model informed by the knowledge of mechanics to make predictions about robot and object motions (Fan et al. [46], Liu [99]). They assume that Newtonian mechanics govern the dynamics of the object and estimate parameters like the mass, inertia, Coriolis component and gravity effects using regression techniques (Mason [106, 105], Lynch [100]). However, most methods make unrealistic assumptions such as quasi-static mechanics, zero slippage and point contacts to make the derivation tractable (Chatterjee [27]). Hence analytic method suffers from the problem of inaccurate prediction. Furthermore, to make predictions, these approaches require an explicit representation of its intrinsic parameters, such as friction, mass, mass distribution, and coefficients of restitution, which are not trivial to estimate (Kopicki et al. [79]).

The second approach is to use techniques of machine learning to build a forward-model. These techniques learn an action-effect correlation either using data obtained from expert demonstrations (Kronander and Billard [81], Huang et al. [65]) or from self experience using trials (Levine and Koltun [92], Kupcsik et al. [86]). These methods do not require explicit mathematical representations of the task, robot, or the objects involved. However, the choices regarding the model representation, learning algorithm and state representation are challenges that need to be tackled.

The third approach which use a combination of both first and second approaches can also be seen in the literature. These forward-models tend to learn the difference between analytical models and the true dynamics of the object. The assumption is that learning these errors will improve the analytical models (Gandhi et al. [53]). The main advantage of these forward-models is low requirement of data since only the difference in dynamics are to be learned.

In contrast to the above works, our work (Chapter 4) can learn dynamics of robot-environment interaction as a forward-model without assuming any prior knowledge. The model is learned online makes no assumptions on the structure of the probability distribution. This makes the method more generalisable and adaptable to new environments.

2.2 Planning push manipulation

Pushing is considered one the most primitive kind of manipulation action and is frequently encountered in robotics (Zito et al. [175]). Object motions occurring due to pushes are hard to predict and difficult to control. Manipulation using push actions is a well studied problem and was initially attempted (Lynch [100], Mason [106, 105], Peshkin and Sanderson [128])

using analytical methods using principles of classical mechanics. The analytical approach models the push effects by using extensive knowledge of the manipulated object properties (e.g. friction, mass, inertia, and centre of mass), robot (e.g. dynamics, kinematics), and environments in contact. However, it is quite difficult to estimate or measure these parameters. Despite these challenges, analytic models have also been extensively employed for push planning (Cosgun et al. [30], Dogar and Srinivasa [40], Mason [103], Stilman and Kuffner [150], Zito et al. [174]).

The difficulty of inaccuracy motivates to learn the forward-models from data or use a hybrid approach (Belter et al. [14], Zhou et al. [173]). Some of the data-intensive approaches are Agrawal et al. [4], Finn et al. [49], Pinto and Gupta [129] and data efficient approaches are Kopicki et al. [79], and Kopicki et al. [78]. However, these approaches fail to predict both uncertainties in dynamics and meta-uncertainty in the object dynamics model, which is useful for robot planning Deisenroth and Rasmussen [34]. The approach developed by Bauza and Rodriguez [11] can predict meta-uncertainty due to a lack of data but seldom use it in planning. A push planning approach put forward in Agrawal et al. [4] uses a Siamese architecture to learn a forward and an inverse model for pushing simultaneously. However, the push actions were described in discrete action space and could not use predictive uncertainty. Finn et al. [49] attempted the push manipulation in a model predictive control schema (similar to us) employing a forward-model learned using auto-encoders yet was not capable of including predictive uncertainty. This work can only perform single push actions which are based on an image input.

A planning framework which relied on analytic models was proposed by Mason [104] which used a known 3D object. A similar idea was utilised for planning push paths for planar objects by Mayeda and Wakatsuki [110]. While this work proposed some strategies to cope up with uncertainties regarding parameters (e.g. friction, mass, centre of mass), yet was not capable of accounting object rotation. Work by Akella and Mason [7] proposes a push planner which is capable of guaranteeing a set of push sequence to take a planar object from a given configuration to any target configuration. The push sequence generated relies on analytic models and can only be executed in an open-loop fashion execution, and there was no mechanism to modify a plan or account feedback. Narasimhan [120] proposed a planner with a set of local task-level feedback controllers to create robust non-holonomic plans for pushing in a 2D world which had a globally convergent behaviour. The local task-level feedback controllers were automatically created and updated from handcrafted difficult to tune simulation models. The scaling of the proposed method to larger dimensions is non-trivial.

Other works (Lynch and Mason [101] and Agarwal et al. [3]) in 2D world planned stable push actions (i.e. push directions that cause the object to remain fixed to the manipulator) on planar objects. The work was based on analytic models and assumed point contact or line contact with the environment and the robot. Similar to this work, a path planning task was attempted by Nieuwenhuisen et al. [122] using only 2D disc-shaped objects in simulation. The work generated an obstacle free push path by taking in a partial section of a viable path and assumed extensive knowledge of the world. The work was extended in De Berg and Gerrits [32] which proposed an approach with a theoretical guarantee to produce the shortest path under the assumption that the robot is in touch with the object at all times. While most of the previous works assumed 2D worlds or planar objects, work in Miyazawa et al. [117] proposed a rapidly-exploring random trees (RRT) based algorithm to plan push actions. The algorithm was able to perform sliding operation on a 3D cuboid with three degrees of freedom under no obstacles in simulation. The algorithm results in a significant variation in computation times since the number of contact locations scales with the size of the object. The work in Cappelleri et al. [26] created an analytic framework for planning push actions under quasi-static assumption and open loop using RRTs. Lee et al. [90] formulated the pushing problem in a 2D simulated world as a hierarchical planning problem. The planning problem was solved in three stages of object contacts, object poses and robot contacts. Another attempt to solve the planning problem in pushing as an open-loop is found in King [76]. This work attempts to solve the push planning under uncertainty in unstructured environments by using an instance of probabilistic planning and gathering information by simulating the world with physics engines. The work in Zhou et al. [172] proposes a polynomial force-motion model for planar sliding. The work was able to generate stable push actions by collecting force-motion data. Later push actions are derived as a solution to an optimisation problem with polygonal approximation to the friction cone and other physics-based constraints (e.g. constraints on mass, density, friction coefficient).

The work described in Chapter 3, an ensemble of Mixture Density Networks (E-MDNs) (Arruda et al. [9]) learnt using adversarial training proposed by Lakshminarayanan et al. [88] can represent meta-uncertainty due to a lack of data. The forward-model comprises of a neural network which learns the parameters of a mixture density to represent the mapping of the current object state, robot action to possible next states. The mixture density represents the effects of the robot's push action. These E-MDNs scale well with training data unlike methods like Gaussian Processes (GP) (Williams et al. [165]) and can represent meta-uncertainty, compared to methods like Gaussian Mixture Regression (Da Silva et al. [31]). Similar to our work, a learned model is employed by Zito et al. [174] and use RRTs to plan push actions; but the push actions generated are not uncertainty averse.

2.3 Learning variable impedance controller

Task specification of a robot can be represented as a *kinematic trajectory*[1]. Simple trajectories can be modelled using curve fitting techniques (e.g. splines, Bézier curves etc) and complicated ones can be recorded from a human expert ("Learning from demonstration" techniques). However, the approach of using kinematic trajectory fails for tasks which involve interactions or application of specific forces with the environment. Such interactions result in stochastic disturbances and hence is challenging to control. A solution is to use controllers in the impedance paradigm. For robotic systems which suffer from noise, disturbances or sensorimotor delays can be controlled using impedance modulation (Mitrovic et al. [116]). But this will also require the knowledge of right impedance parameters.

One approach to derive impedance parameters is to record multiple demonstrations from a human expert and vary the stiffness as a function of the variability in the collected trajectories (Calinon et al. [22], Kormushev et al. [80], Paraschos et al. [127]). The main disadvantage of these methods is that the kinematic variations are related to the stiffness profile, which may not be the case always as shown by Li et al. [93]. These approaches assume that the teacher conveys the right stiffness information. Instead of deriving impedance parameters from kinematic data, another related way is to supply them through expert demonstration via a haptic interface (Medina et al. [112]). Haptic device allows recording of interaction forces used by the expert to do the task. However, the haptic device used in such approaches are specifically designed, expensive and often inaccessible to others (Kronander and Billard [81]).

A related approach to derive time-varying stiffness profile from demonstration is by using the Linear Quadratic Regulator (LQR) control strategy (Medina et al. [111]). Initially, the kinematic demonstrations are used to compute the inverse covariance matrices. These covariance matrices represents trajectory variability and are penalised by the optimiser to compute optimal stiffness values. A related approach uses the minimum intervention principle based controller implemented via the LQR (Calinon et al. [24]) to derive impedance parameters from demonstrations. Alternately, demonstrations are used as priors (Lee et al. [89]) or as post-estimation projection onto admissible stiffness matrices (Rozo et al. [135]). All these works derive time dependent impedance matrices as opposed to our state dependent impedance parameters. Another core difference is that the impedance adaptation proposed in this thesis is independent of the kinematic demonstrations given and in this regard is similar to the approach of Kronander and Billard [81].

[1] A time stamped sequence of end-effector or joint positions

Optimal control (Mitrovic et al. [116], Braun et al. [17], Garabini et al. [55]) and the closely related reinforcement learning (Stulp et al. [153], Buchli et al. [19], Calinon et al. [23]) are used to learn variable impedance control policies and produced impressive results. However, they all rely on handcrafted task-specific cost function thereby demanding explicit task knowledge and robot dynamics (Schaal et al. [140]). These requirements further make the transfer of a learned policy to a new system difficult as they are generally defined in the joint space of the robots. Moreover, such a transfer also demands cost function redesign and retraining of the system. This relearning will be particularly damaging since dangerous contact forces can arise in interaction tasks (Burdet and Nuttin [20]).

Other learning attempts (Asada [10]) included supervised learning, which used neural networks to provide non-linear compliance by learning a mapping from end-effector force to velocity. However, such approaches demanded the existence of large noise-free training data sets and produced deterministic outputs. Similar learning based approaches were followed in (Kalakrishnan et al. [71, 69], Braun et al. [18], Kupcsik et al. [84], van Hoof et al. [162]) presented stiffness profiles as a time series or as a task-specific blind policy.

Another interesting approach which does not demand the implementation of a handcrafted cost function is via inverse reinforcement learning originally proposed by Abbeel and Ng [2]. This approach was used for transferring impedance modulation in some manipulation tasks (Howard et al. [63]). However, most tasks pursued using this approach required explosive movements (e.g. throwing a ball) and did not involve close interactions with the environment (Haddadin et al. [59], Stulp et al. [153]). In contrast, our approach is defined in the task space of the robot and does not demand any robot-specific knowledge. The framework proposed is able to deal with continuous interaction tasks. A related work is presented in the fifth chapter of Kupcsik et al. [84] which put forward an offline trained state dependent stiffness controller. But this approach was utilised only in peg-insertion tasks and controlled single axis stiffness. Our work proposes a minimal tuning, multi axis online training framework which is used in a wide variety of tasks.

In essence, the method proposed in this thesis has similar behaviour to adaptive control methods. In adaptive control, methods achieve high performance by changing control parameters and without excessive dependency on system models. The modification of control parameters uses high-gain learning rates which often results in system instabilities (Yucelen and Haddad [171], Dydek et al. [43]). In our method, we start with a system model learned from a few demonstrations to predict interaction forces. However, these predictions have high deviations from the actual values experienced. To compensate for the error in predictions, we adapt the control parameters (similar to adaptive control methods) as per the error. In contrast, our approach has an online learning module which modifies the interaction model

of the task. The improvement of the model results in better predictions and subsequently reduces the need for high-gain control parameter adaptation. The learned task model can be used to initialise the forward-model in our framework in other similar tasks. Our results show that such initialisation helps in faster learning of the models.

There have been several works in motor control literature which strongly provides evidence (Osu and Gomi [126], Gomi and Osu [57]) for the existence of varying impedance behaviour in human beings. The impedance modulation in human beings is achieved by co-contraction of muscles (Rosenbaum [134]). Studies by Burdet et al. [21] suggest that motivation for modulation of stiffness by humans is to deal with unstable dynamics. Other studies by Selen et al. [143] suggest it could be due to the presence of sensorimotor noise. Existence of task-based impedance adaptation in humans is supported by evidence from Franklin et al. [52], Takahashi et al. [155], Wang et al. [163]. Ideas of these studies are utilised in the controller suggested by Yang et al. [170] and further extended by Ganesh et al. [54]. The controller provided a constant feed-forward force for predictable perturbations and increasing stiffness in directions of unpredictable perturbations. Further, studies performed by Ajoudani et al. [6] report human assisted stiffness modulation via tele-operation outperforms constant or low stiffness behaviour of the slave robot for a given task.

Therefore, instead of learning to vary impedance parameters directly, the proposed framework varies them by learning a forward-model of the task (Chapter 4). It is also observed that the human motor system uses predictive models for anticipating the action effects (such as reaction forces) and motor actions on sensory data (Flanagan et al. [51], Johansson and Cole [68], Kawato [74]). These predictions are used for different purposes such as feed-forward control, motor system coordination, action planning and monitoring.

2.4 Control challenges

Research in robot control was mostly related to high precision movements in free space. However, such schemes provide accuracy to follow unobstructed trajectories but fail to facilitate control when contact interactions are required. The main challenge arises due to the conflicting nature of position and force. The robot achieves position accuracy by being stiff, but trades it off for being compliant to external forces.

There are several approaches in classical control like hybrid force control (Salisbury [137]), parallel force control (Chiaverini and Sciavicco [29]) designed from a manipulator perspective to achieve varying stiffness control and impedance control (Hogan [62]). Limitations of these approaches include access to accurate system dynamics model and precise feedback sensors.

Other approaches to design varying stiffness control from the object perspective like Schneider and Cannon [141], Wimböck et al. [167], Li et al. [94]. Most of these approaches are specifically designed for grasping and are based on accurate analytic models of the object. It is challenging to precisely perform a system identification of the object while being manipulated by a robot. Learning based approaches like Kronander and Billard [81] require substantial external hardware to collect demo from the humans to encode and replicate the stiffness variation in tasks.

The approach followed in our work (Chapter 3) is mostly similar to the some RL-based formulations such as Kalakrishnan et al. [71], Buchli et al. [19], Kalakrishnan et al. [69], Braun et al. [18], Howard et al. [63], Kupcsik et al. [84] and van Hoof et al. [162]. These works either represent the stiffness profiles as a time series or as a task specific blind policy. In contrast, our work represents the stiffness adaptation as a state dependent entity instead of being time dependent. By defining the stiffness controller as a state-dependent property, we propose that the same module can be re-used in other tasks involving similar features.

2.5 Preliminaries

We need to understand some preliminaries before getting into the details of the thesis. Following sections give short tutorials on different techniques and methods used in this thesis. Introduction to the task space control is given in Section 2.5.1 which is later used in Chapter 4. We use forward-models learned using probabilistic approaches and Section 2.5.2 gives a brief on the same. The push manipulation research proposed in Chapter 3 uses optimal control (Section 2.5.3 strategies such as the *Path Integral* approach (Section 2.5.4).

2.5.1 Task space control

Human motor control and behavioural studies (Saltzman and Kelso [138], Scholz and Schöner [142], Todorov and Jordan [161], Schaal and Schweighofer [139], Mistry et al. [115]) show that human manipulation behaviour has features equivalent to that of compliant task space control in robotics. We tend to control the stiffness in relevant degrees of freedom while keeping others compliant. This property makes them robust to stochastic disturbances and can perform safe interactions with the environment. The main motivation for developing task space control strategies for robot manipulators is to achieve human arm like performance.

The task space control paradigm aims to produce correct motor commands depending on task goal and robot configuration. The problem of task space control uses the manipulator

equation (Equation 2.1) which is formulated using the rigid body assumptions (Featherstone [47]).

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \tau - \mathbf{J}_C^T \mathbf{F}_{ext} \tag{2.1}$$

where $\mathbf{q}$ is the (generalised) joint angle vector, $\dot{\mathbf{q}}$ is the joint velocity vector, $\ddot{\mathbf{q}}$ is the joint acceleration vector, $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is the Coriolis vector, $\mathbf{g}(\mathbf{q})$ is the gravity vector, and τ is the joint torque vector. $\mathbf{J}_C$ is the contact Jacobian and $\mathbf{F}_{ext}$ is the external force in the Cartesian space. Here we are assuming $\mathbf{J}_C = \mathbf{J}$ the end effector Jacobian. This equation is defined in the joint space of the robot and needs to be translated to its equivalent in the task space for implementing task space control. After rearrangement of terms (interested readers can refer to Appendix A), the task space equivalent of Equation 2.1 is given by Equation 2.2.

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{g}(\mathbf{x}) = \mathbf{F} - \mathbf{F}_{ext} \tag{2.2}$$

where, $\mathbf{x},\dot{\mathbf{x}},\ddot{\mathbf{x}}$ is end-effector position, velocity and acceleration respectively in the Cartesian space, $\mathbf{M}(\mathbf{x})$ is the end-effector inertia [2], $\mathbf{C}(\mathbf{x},\dot{\mathbf{x}})$ is the task space equivalent of Coriolis component, $\mathbf{g}(\mathbf{x})$ is equivalent gravity component, $\mathbf{F}$ is force required in the task space to move the robot and $\mathbf{F}_{ext}$ is the interaction force felt.

The task space control for free space movement deals with the design of $\mathbf{F}$ using either force based control (Khatib [75], Featherstone and Khatib [48]), acceleration based control (Hsu et al. [64], Senda [144]) or velocity based control (Nakamura and Hanafusa [118], Liegeois [96]). The exact control laws of these formulations can be read in Nakanishi et al. [119].

However, when there is an interaction force $\mathbf{F}_{ext}$ acting on the robot, the control laws by different schemes mentioned above are insufficient. The external force affects the motion of the arm. For example, consider when robot arm is moving down to a target pose and there is a non-penetrable surface in its path. The robot on reaching the surface will apply more force on its tip to move towards the target. Such unsafe behaviours may damage the robot and should be avoided by adding compliance behaviour to the arm motion.

A way to add compliance behaviour is by compensating $\mathbf{F}_{ext}$ either via the modification of $\mathbf{F}$ using impedance control scheme (Hogan [62]) or by adjusting the arm posture in the null-space of the robot without affecting the current (using the available task irrelevant control directions) task state. Our work described in Chapter 4 proposes an approach to achieve compliance for tasks that involve continuous interaction with the environment.

[2] equivalent inertia of the arm felt at the end-effector

2.5.2 Probabilistic modelling

Consider a robot in an unknown environment. All capabilities required for the survival of the robot in such an environment may not be trivial to program manually. If the robot has to behave intelligently, it should have the ability to sense the environment, make beliefs about its world propositions and infer decisions based on them. For example, in case of a manipulator moving an object, this proposition can be about the object's location, its own end-effector pose etc. The beliefs can be represented mathematically from data collected by the robot through its sensors. However, inference from data is intrinsically uncertain. Hence, we use probabilistic models to represent these beliefs. These representations can include all forms of uncertainty and noise (Ghahramani [56]). The models can either make assumptions about its structure (called parametric methods) using domain-specific knowledge or can be totally data centric (called non-parametric methods) (in these approaches the model structure is unconstrained and is learned from data). This thesis uses parametric representations and is illustrated by the following example.

Consider a data set $\mathbf{D} = [\mathbf{x}_n, \mathbf{y}_n]$ for $n = 1 : N$ where N is the number of data points, $\mathbf{x}_n$ is the input vector and $\mathbf{y}_n$ is output vector. Let us assume that there exist mapping between $\mathbf{x}_n$ and $\mathbf{y}_n]$. The mapping can be represented mathematically as a function (f) as in Equation 2.3.

$$\mathbf{y}_n = f(\mathbf{x}_n|\theta) \tag{2.3}$$

where, θ is the parameters which determine the model and is learned from data $\mathbf{D}$. One example of f is the multivariate Gaussian distribution.

$$f(\mathbf{x}_n|\theta) = \mathcal{N}(\mathbf{x}_n|\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} exp(\big[-\frac{1}{2}(\mathbf{x}_n-\mu)^T\Sigma^{-1}(\mathbf{x}_n-\mu)\big]) \tag{2.4}$$

Here, θ is the mean and covariance of the μ, Σ of the distribution. We use different algorithms to estimate the parameter θ from data $\mathbf{D}$ such that:

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \;\; \mathbf{y}_n - f(\mathbf{D}|\theta) \tag{2.5}$$

where, $\hat{\theta}$ is the estimated value of the parameter θ using the algorithms Expectation-Maximisation(EM) either offline (North and Blake [123]) or online (Cappé [25]). The learned parameter $\hat{\theta}$ is best parameter which can explain the data. More complicated probability distributions are modelled by using a mixture of multivariate Gaussian (Equation 2.6)

as we have used in this thesis.

$$f(\mathbf{x}_n|\theta) = \sum_{i=1}^{K} \omega_i \mathcal{N}_i(\mathbf{x}_n|\mu_i, \Sigma_i) \tag{2.6}$$

where, K denotes the number of Gaussian kernels in the distribution, ω_i is the individual contribution (weight) of i^{th} kernel, parametarised by mean (μ_i) and covariance (Σ_i). For these mixture distributions, the weights ω is also estimated along with the mean and covariance using the same EM algorithm. However, when the data $\mathbf{D}$ is not completely available beforehand, the learning of the mixture density can be performed incrementally (Engel and Heinen [44], Ahmad [5]) as each datum comes in. Chapter 4 uses and describes such a learning technique (Section 4.2.2) to learn the dynamics of some robot-environment interaction tasks (Section 5.2).

An alternate representation to learn $f(\mathbf{x}_n|\theta)$ from large data sets is by using *Neural Networks*. The network may contain layers of neuron (connected each other) depending on the complexity of the learning problem. This approach learns a non-linear mapping (Equation 2.7) of the input to the output using *back-propagation* algorithm (Rumelhart et al. [136]).

$$\mathbf{y}_n = f(\mathbf{b} + \sum_{n=1}^{N} \mathbf{w}_n^T \mathbf{x}_n) \tag{2.7}$$

where, $\mathbf{w}_n$ is the weight of the input $\mathbf{x}_n$, $\mathbf{b}$ is the bias, f is the *activation function* [3] specified.

Initially, the parameters $\mathbf{b}$ and $\mathbf{w}_n$ are randomly initialised, and produce an arbitrary output $\hat{\mathbf{y}}_n$. The goal of the back-propagation algorithm is to adjust the weights of the connections in the network in an iterative manner to minimise the difference between the true output $\mathbf{y}_n$ and current network output $\hat{\mathbf{y}}_n$. The weight adjustments help the internal neuron layers of the network to capture important features of the data $\mathbf{D}$. The regularities in the data is captured by the interactions of these neurons (Rumelhart et al. [136]). Typically the neural networks are used for learning deterministic outputs. Currently, they are increasingly getting used to learn the parameters of the distribution such as mixture of Gaussian (Equation 2.6). Chapter 3 uses and covers details of such a model to learn dynamics of an object (Section 3.2.1) being pushed on a table.

2.5.3 Stochastic Optimal Control

Classical optimal control (COC) deals with finding a control law for a deterministic dynamical system by minimising a cost function. The framework optimises the cost function over a

[3] The linear or non-linear function which maps the the scalar value ($\mathbf{b} + \sum_{n=1}^{N} \mathbf{w}_n^T \mathbf{x}_n$) to the feature space. Types of activation functions can be read in Nwankpa et al. [124]

fixed or varying time horizon (Kappen [73]). The optimisation is performed chiefly in two ways. The first way is to use the *Hamilton-Bellman-Jacobi* (HJB) formulation which solves the problem using *dynamic programming*. The second is the *Pontryagin's Minimum Principle* (PMP) which solves the problem using standard techniques of calculus (by formulating it as ordinary differential equations)(Kirk [77]). Both techniques try to find a globally optimal solution which minimises the cost. It is to be stated that the lack of consideration of stochastic disturbance on system dynamics is a notable drawback of this framework.

Stochastic optimal control (SOC) on the other hand deals with uncertainty in system dynamics due to inaccurate sensor-environment models, inherent noise in the resultant state or imperfect actions. SOC also creates a similar agent to COC find an optimal control law for a dynamical system which is subjected to noise. Inclusion of noise in the HJB formulation is straight-forward compared to the PMP formulation. However, this class of problems are intractable since the formulation requires discretisation of the state-space. Converting higher dimensional state-space to their discrete equivalent is impractical. Reinforcement Learning (RL) paradigm also faces the same problem (Kappen [73]). The solution to stochastic optimal control can be viewed as a weighted mixture of sub-optimal solutions (Kappen [73]). This weighting is sensitive to problem-specific features, noise and time horizon (since these parameters affect the cost function).

Optimal control can be seen as an extension of Dynamic Programming to continuous sequential decision processes (Bellman [13]), via Differential Dynamic programming. As a first step, the dynamics of the system to be controlled is defined using the Hamilton-Jacobi-Bellman partial differential equation (PDE). Later, this formulation is numerically solved by moving backwards in time, provided that the initial and final system configuarations are specified (Kappen [72]). For following a trajectory, the control command sequence can be obtained by optimising the sum of each step costs (called the running cost) along it.

Solving the (in general) non-linear HJB equation to find an optimal sequence of control commands is non-trivial, unless certain assumptions are made regarding the form of the individual step cost and dynamics. For the particular case assuming linear dynamics, quadratic (state and control) costs and linear dependence on zero-mean, white noise can be solved using the standard Linear Quadratic Regulator (LQR) method in closed form (Stengel [149]). A variation to this involving multiplicative noise was solved in (Todorov [160]). Both these solutions are straightforward for linear systems (Li and Todorov [95]), but non-trivial for non-linear systems. However, the assumptions made in the problem formulation are quite restrictive for robotic applications (Kappen [73]).

The class of problems with non-linear dynamics, arbitrary state costs and quadratic control costs are solvable using sampling-based approaches (Kappen [72]). The formulation

make use of the exponentiated cost-to-go function. This usage leads to a Partial Differential Equation (PDE) formulation which can be solved using the Path Integral (PI) based approach.

2.5.4 Path Integral applications for control

Suppose we define a trajectory as being a sequence of states $\mathbf{x}_t$, control commands $\mathbf{u}_t$ at time step t. Thus, let $\mathbf{s}_{t,k}$ be a convenient tuple, such that $\mathbf{s}_{t,k} = (\mathbf{x}_{t,k}, \mathbf{u}_{t,k})$. Then, using k as indices for trajectories, and i as indices for discrete timesteps, the optimal control problem starts with defining a cost function for a trajectory k starting from timestep i until T, i.e. $\tau_{i,k} = [\mathbf{s}_{i,k}, \mathbf{s}_{i+1,k}, \mathbf{s}_{i+2,k}, ..., \mathbf{s}_{T,k}]$:

$$\mathbf{S}_i = \mathbf{S}(\tau_{i,k}) = \phi_T(\mathbf{x}_T) + \sum_{t=i}^{T-1} r_t(\mathbf{x}_t, \mathbf{u}_t) \tag{2.8}$$

where, r_t is the immediate cost function and ϕ_T is the final cost function.

Our aim is to find a *policy* [4] which optimises the above cost function. At each state, an action specified by a policy takes the system to a state in its state-space. An optimal policy however specifies an action which takes the system from the same state to a state which has the least *value function* [5]. Mathematically, the value function for a state $\mathbf{x}_i$ can be defined as:

$$V(x_i) = \min_{u_{i:T}} E_X[S_i] \tag{2.9}$$

At any state $\mathbf{x}_i$, the aim is to find a set of control commands $(\mathbf{u}_i, ...\mathbf{u}_T)$ that takes the system to the goal through intermediate states which possess low value functions. This can be achieved by minimising expected costs from each state $\mathbf{x}_i$. However, direct solving of the non-linear HJB formulation are non-trivial and in many cases non-tractable. A solution to this issue was proposed by Kappen [72] using the Feynman-Kac theorem. In this formulation, the non-linear HJB is converted to a set of linear PDEs which can be solved using forward sampling of trajectories (Theodorou et al. [157]). The added advantage of this approach is the ease of using arbitrary state costs which need to be differentiable. Hence, it applies to a large range of non-linear dynamical systems such as in robotics. Some of the recent works (Chebotar et al. [28], Williams et al. [165, 164]) uses this path integral based approach to solve a wide variety robotic control problems.

Our work in Chapter 3 also makes use of the path integral framework. Here we apply a modified version of the model predictive path integral control algorithm proposed by

[4] A function which maps from each state to corresponding actions.

[5] The value function for a state is defined as the minimum cumulative cost the agent can obtain from a state, if it proceeds optimally from that state to the goal.(Kirk [77])

Williams et al. [165] for push planning. We show the ability of the system to plan uncertainty averse pushes in simulation and to demonstrate general push plans on the real robot.

Chapter 3

Push planning under uncertainty

This chapter describes the initial thesis contribution and is a part of an equally contributed collaborative research presented in Arruda et al. [9] titled *Uncertainty Averse Pushing with Model Predictive Path Integral Control* . In this work, Arruda. E proposed a Ensemble of Mixture Density Networks (Ensemble-MDNs) (a short description of which is available in Section 3.2.1) to learn the forward-model of the object which was used by the model agnostic stochastic planner proposed in this Chapter. The thesis contribution described in this chapter details the development of a stochastic planner which can use any forward-model to perform push planning. The planner is capable to use forward-models with knowledge of its predictive uncertainty (such as Ensemble-MDNs) to perform risk-averse action planning.

Firstly, we describe the motivation of the problem through an example (Section 3.1) followed by details of the proposed solution (Section 3.2). The framework developed is validated through three hypotheses by conducting three experiments (Section 3.3). Results of the experiments are details in Section 3.4 followed by a discussion (Section 3.5).

3.1 Motivation and problem statement

Consider a robot pushing an object on a table towards a target pose. For that, we need to design a *planner* [1] which can help the robot to find a sequence of push actions to move the object. Executing the push sequence can make the object achieve several intermediate poses before reaching the target pose. At each intermediate pose, the planner has to find a push action which moves it closer to the goal. Each push action involves finding its direction and location on the object's surface. Subsequently, the robot applies a certain amount of force in the push direction at the push location found. The force required to move the same

[1] An agent or an algorithm which finds the right sequence of actions to push the object towards the target pose is called a planner.

object may vary depending on the target pose and surface friction. While executing the push sequence, there might be cases when a push action is inadequate, and the object does not move as planned. In such cases, the planner has to adjust the subsequent pushes by taking the feedback (about the resulted object pose from the previous push action) into consideration.

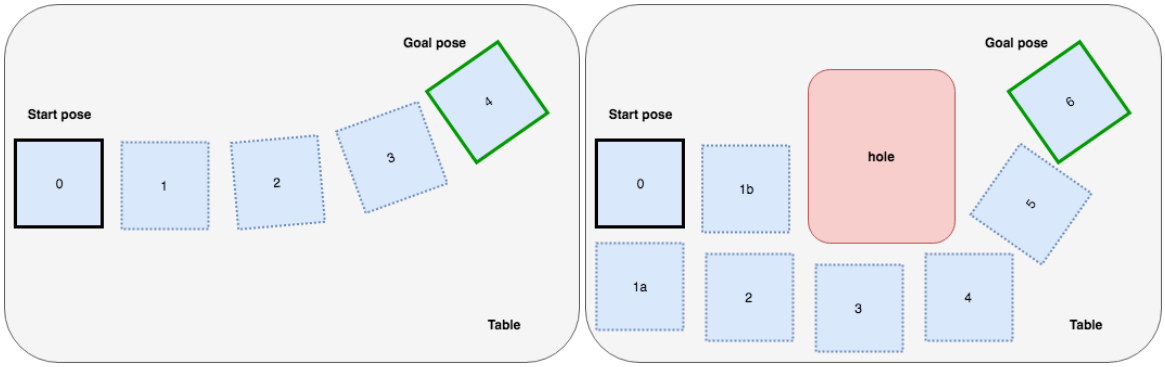


Figure 3.1: Pushing a box on a table. The start pose of the box is given in black border and the target pose is given in green border. The boxes with black doted border are intermediate poses when the box is pushed from the start pose to the goal pose. **Left:** Pushing a box using a sequence of push actions. From the given start to goal pose one obvious set of intermediate states from corresponding push actions is shown $[0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4]$ **Right:** Pushing a box between same start and goal location avoiding the hole at the centre of the table. Two possible solutions of the intermediate object poses are $[0 \rightarrow 1a \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6]$ or $[0 \rightarrow 1b \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6]$

Figure 3.1 gives an insight into the problem of finding the push sequence to move the object. Image on the left side of the figure shows pushing the box from a given start pose to the goal pose along the shortest path. This task is straightforward since every region on the table is *known*[2]. The figure on the right shows pushing the object to the same target pose by avoiding an *unknown* [3] region at the table's centre. This unknown region could be a hole, as shown in the figure or a slippery region which needs to be avoided.

The choice of the push action is influenced by object's behaviour and the target pose. Actions taken with complete knowledge of the object's behaviour will be less risky. On the other hand, actions that take the object to areas of high uncertainty (unknown region) may result in potential failure of the task. Hence it will be ideal to stay closer to the known regions (less uncertain areas) while moving towards the goal. For example, the image to the right of the Figure 3.1, there exist two paths towards the goal. However taking the push action at *pose 0* towards *pose 1a* will be less riskier than taking a push action which lead the object to *pose 1b*. Being in *pose 1b* has a higher chance of moving towards the unknown region than

[2]Known region is a space in which we have certain knowledge about the object's behaviour under an action.
[3]Unknown region is where the planner is highly uncertain about the behaviour of an object under an action.

pose 1a. So a good planner should choose the path [0 → 1a → 2 → 3 . . .] instead of the path [0 → 1b → 2 → 3 . . .].

Finding such actions require the planner to know the object's behaviour (under a push action) on the surface of the table. This object's behaviour is given by its *forward-model* which can be either analytic or learned from data. The model can be used to forward simulate the effects of various push actions on the object. The planner uses the knowledge of these effects to choose the best action.

A forward analytic model typically requires the knowledge of different object properties like mass, inertia and environment properties like friction. Estimation of these parameters accurately is cumbersome. Apart from this requirement, analytic models often make unrealistic assumptions such as point contacts to be present between the object and the table's surface (Arruda et al. [9]). Similar problems also exist when physics engine based models are used for simulation (Kopicki et al. [79]). A better alternative is to learn a forward-model from data. Such a model learns a functional representation between an input (current pose, a push action) and an output (resulting pose). The data is collected by measuring the start and end pose of the object under random pushes by the robot. Each datum consists of the start pose, the end pose and the push action applied by the robot. The data collected is used to train a forward-model which can be used for predicting object's state under an action.

The state predicted by a object's forward-model (analytic or learned) simulated under an action may not be the same as when applied to the real object. This difference between the prediction and the true value (called as uncertainty in the forward-model) arises due to several factors such as; (1) approximations and assumptions taken in an analytic model, (2) meta uncertainty due to limited data in a learned model or (3) inherent uncertainty in object-table interaction dynamics. The highly uncertain regions on the table's surface posses high risk. These risky areas can be avoided if the forward-model provides measure of the uncertainties associated with its prediction. In short, our planner has to find push actions which avoid uncertain regions while moving the object towards the target pose. Mathematically, such 'intentions' of the planner are defined using a cost function. A cost function provides a quantified measure on the merit of a push action by mapping values of one or more variables associated with it to a single scalar value. For example, consider the following equation.

$$r(\mathbf{x_i}, \mathbf{u_{i-1}}, \overset{*}{\sigma_i}) = f_1(\overset{*}{\sigma_i}) + f_2(\mathbf{x_i} - \mathbf{x_{goal}}) + f_3(\mathbf{u_{i-1}}) \tag{3.1}$$

where, f_1 is a function of $\overset{*}{\sigma_i}$(uncertainty associated with the pose x_i) which quantifies the extent of the uncertainty, f_2 is a function of x_i (current pose) which quantifies how good this pose is with respect to the x_{goal} pose and f_3 is a function of u_{i-1}(the last chosen push action resulted in x_i) which quantifies how expensive was the last action.

All the functions f_1, f_2 and f_3 gives a scalar number as output. The sum of result of f_1, f_2 and f_3 gives the total goodness of taking the action u_{i-1}. More specific details of the cost function designed for our push task can be read in the Section 3.2.2.

Often having a forward-model is insufficient to select the right action due to problems like (1) unforeseen disturbances in the environment, (2) errors in the model due to approximation, (3) inaccuracies due to insufficient data, and (4) drift in the model output due to change in environment and object properties due to wear and tear. Thus a push action from a state (pose) results in stochastic (may even be multi-modal) outcome. Hence it is essential to develop a framework which can deal with a stochastic outcome.

This chapter explains such a framework which can perform robust planning for robot push manipulation using a learned forward-model. The forward-model used can predict the resulting state (from a given state and action) and provide the measure of the associated uncertainty. We explore the possibility of using learning methods (e.g: Gaussian Process Regression, and an Ensemble of Mixture Density Networks that give estimates of the uncertainty in their predictions) and stochastic planners. The stochastic planner (Model Predictive Path Integral - MPPI) proposed uses these learned models to plan a sequence of push actions to move the object towards the target pose. Thus, our planner uses non-differentiable cost functions to complete the pushing tasks robustly with respect to the predictive uncertainty of the forward-models.

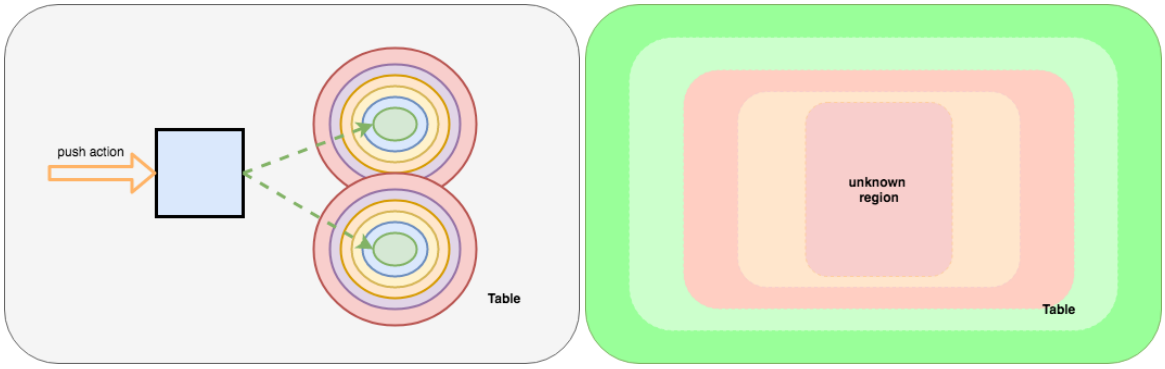


Figure 3.2: Table uncertainty. **Left:** Stochasticity when a push action is taken **Right:** Uncertainty of the table

Using the planner described in Section 3.2, we experimentally evaluate two versions of the approach for push manipulation. The first version (Section 3.2.2) makes uncertainty-averse push actions by jointly minimising uncertainty and goal costs, which makes it difficult to find optimal push actions. The second version (Section 3.2.3) initially creates an uncertainty averse path between the start and goal pose and then finds push actions to follow this path.

The method is also validated on a real robot (Baxter) (Section 3.3) to show that learned models with stochastic planners can outperform traditional methods (Section 3.4).

3.2 Proposed solution

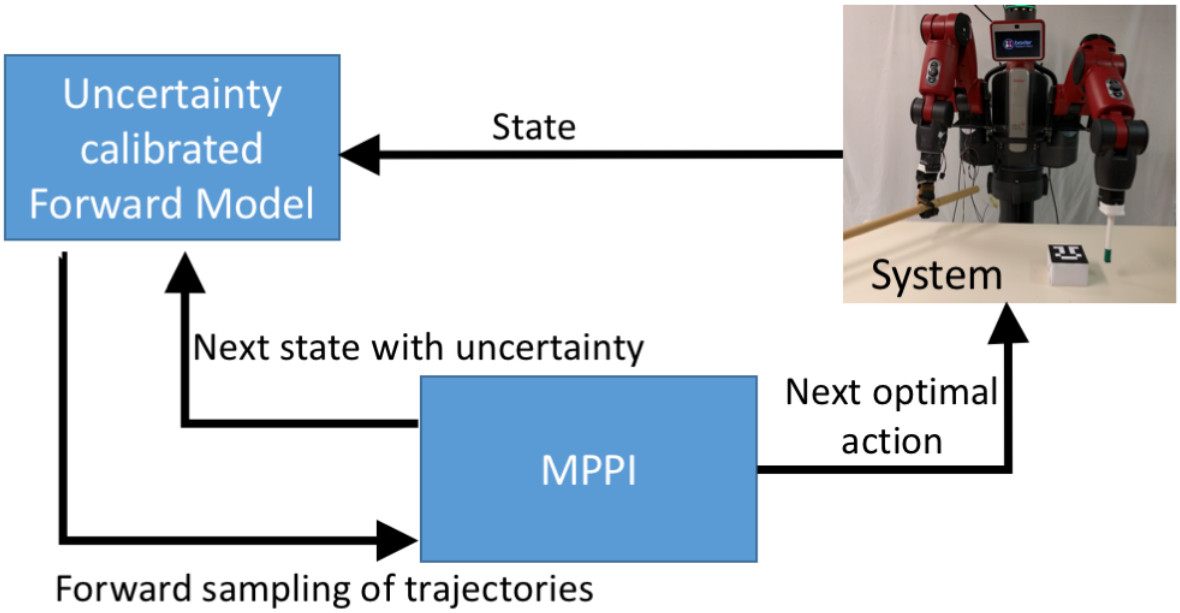


Figure 3.3: Overview of the proposed solution. The model predictive path integral based planner uses an uncertainty calibrated forward-model. Optimal push action is chosen in and executed on the robot. After each push action, the pose of the object is fed back to the model.

Given a target pose, our problem is to figure out a way to plan a sequence of risk averse push actions to move the object. It is essential to know the risk associated with the resulting object pose trajectory to find such a push sequence. A way to quantify the risk is to sum the uncertainty associated with constituent poses (individual states) in the trajectory. From the start pose, the effects of different pushes can be computed by simulating the forward-model. The planner can avoid high risk states by using the result of these simulations.

However, planning performed using a forward-model with modelling errors or unforeseen disturbances can produce unintended consequences during execution. Hence open loop execution of a plan may not generate the intended object trajectory. This problem can be solved by incorporating feedback into planning. In our control architecture, the push sequence is computed over a finite horizon by the optimisation of a cost function in an iterative manner. The sequence of control (push) is calculated by predicting future states using the forward-model. After calculation, the first command of the computed control

(push) sequence is executed and feedback of the system is incorporated. The feedback helps to rectify the possible errors in the predictive states due to inherent non-linearity or approximations of the forward-model. Subsequently, iterative optimisation is re-started to find the next set of control commands [4].

Two methods are proposed and tested to solve the problem of finding the push sequence. The first method will push the object towards the goal in a risk averse fashion; that is, the trajectory of the object moving and the push required to keep it on the trajectory is generated simultaneously. This is done by optimising the cost function specified in the Equation 3.11. This optimisation is similar to finding a solution which can satisfy two constraints (moving towards the goal and being risk averse) simultaneously. More details of the approach is described in the Section 3.2.2 and the algorithm used is described in Algorithm 1.

The second method generates a risk averse trajectory upfront and then plan a sequence of push actions to follow it. The advantage of this approach is that the planner does not need to optimise the combined cost function specified in Equation 3.11. Instead, the planner can just optimise the Equation 3.14 initially to generate risk averse trajectory. Now this trajectory generated can be used by the planner to generate the push sequence by optimising Equation 3.15. The breaking of the search problem into two phases makes the search easier and more directed since, the generated trajectory acts as a heuristic to the planner for finding the push actions. The generation of trajectory samples can be around the initial chosen trajectory (this could be randomly generated as well or given by the programmer) for finding optimal push actions. The cost function are used in two stages separately. The first phase uses only uncertainty penalisation while the second phase uses only goal penalisation. Details of this approach and the algorithm 2 is given in Section 3.2.3.

3.2.1 Forward-models with predictive uncertainty

Capturing of uncertainty in predictions involve learning of probability densities. Single and multi-modal probability densities for real-valued data can be learned in various ways. Popular choices are non-parametric models such as Kernel Density Estimators (KDE), Gaussian Processes (GP) or parametric models such as Gaussian Mixture Models (GMMs). However, not all of these methods have the capability to capture true uncertainty in predictions. Gaussian Processes, for example, provide an effective and theoretically clean tool to make uncertainty aware predictions (Rasmussen and Williams [131]). The problem with Gaussian processes are their inability to scale to high dimensional spaces or large data sets. Gaussian Mixture Models though are scalable to large dimensions, they are highly limited in their

[4] This way of closed loop planning technique is called *Model Predictive Control* or *Receding Horizon Control* in literature.

capability to extrapolate uncertainty to regions outside the training data. This shortcoming of the GMMs can be solved by using ensemble of them together for predictions. Such a model called Ensemble Mixture Density Networks (E-MDNs) is proposed by Arruda et al. [9]. E-MDNs is an ensemble of neural networks in which each network learns parameters of a multi-modal Gaussian mixture probability distribution. The use of neural networks for implementation helps in using large data sets for training.

Mixture Density Networks (MDNs) first proposed by Bishop [15] showed how they are effective in modelling arbitrary densities. In such models, the network outputs the parameters of a Gaussian mixture model (if the choice of the mixture components are Gaussian) conditioned on a suitable choice of input vector $\mathbf{h} \in \mathscr{R}^n$, thus modelling arbitrary multi-modal densities as defined in Equation 3.2.

$$p_\theta(\mathbf{x}_{t+1}|\mathbf{h};\theta) = \sum_{k=1}^{K} \pi_k(\mathbf{h};\theta)\mathscr{N}(\mathbf{x}_{t+1}|\mu_\mathbf{k}(\mathbf{h};\theta), \sigma_k^2(\mathbf{h};\theta)), \tag{3.2}$$

where $\pi_k(\mathbf{h};\theta)$, $\mu_k(h;\theta)$ and $\sigma_k(\mathbf{h};\theta)$ are the network outputs which form the parameters of the mixture. A soft-max layer is used to represent π_k, to guarantee generation of valid parameters for the mixture by the network (Bishop [15]), such that $\sum_{k=1}^{K} \pi_k(\mathbf{h};\theta) = 1$, whereas an exponential layer can make sure that σ_k is positive definite, and finally μ_k can be represented as a linear combination of hyperbolic tangent activation functions. Further considerations on practical implementation of MDNs are covered extensively in Bishop [15], Graves [58].

The data point for learning a forward-model is defined as $\mathbf{h} = (\mathbf{x}_t, \mathbf{u}_t)$, where $\mathbf{x}_t$ is the state at time step t and $\mathbf{u}_t$ is the action taken at that time. The predictive uncertainty of the learned forward-model can be estimated by forming an ensemble of such models (Lakshminarayanan et al. [87]). If an ensemble is composed of M members, the final model can be written as:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{M}\sum_{m=1}^{M} p_{\theta_m}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t; \theta_m) \tag{3.3}$$

The statistics of interest which can be computed from the ensemble are the mean prediction and the variance, which trained accordingly can reflect the predictive uncertainty of the model (Lakshminarayanan et al. [87]), and are given by:

$$\overset{*}{\mu} = \frac{1}{M}\sum_{m=1}^{M} \mu_{\theta_m} \tag{3.4}$$

$$\overset{*}{\sigma}^2 = \frac{1}{M}\sum_{m=1}^{M} (\sigma_{\theta_m}^2 + \mu_{\theta_m}^2) - \overset{*}{\mu}^2 \tag{3.5}$$

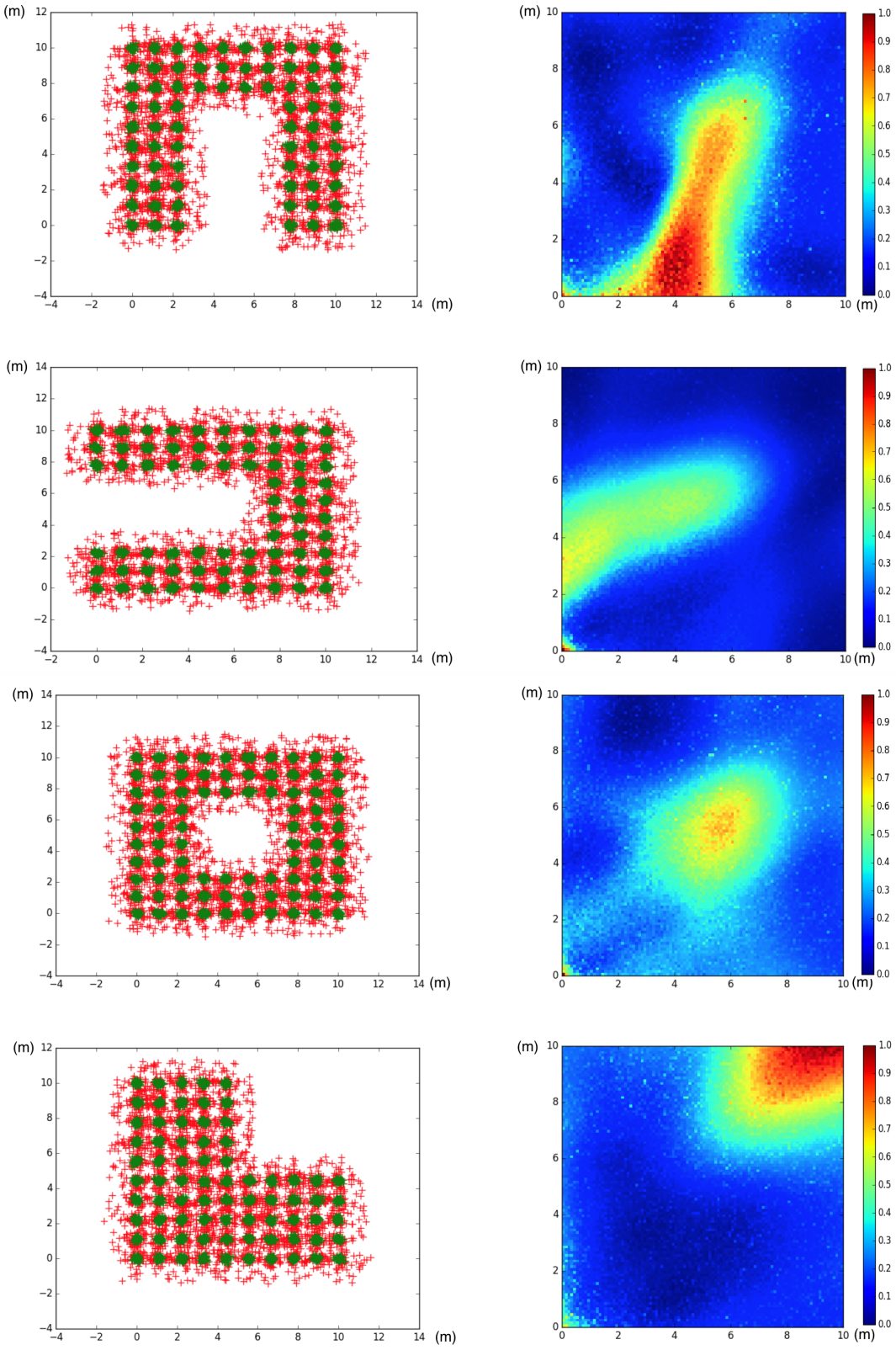


Figure 3.4: The physics engine Box-2D is used to generate some toy push data sets shown on the left. Green crosses indicate start locations and red ones indicate end locations. Lesions are created in data generated (the gaps in the data set) and is used for training the forward-model. The right indicate the heat map of the predictive uncertainty of the forward-model in different regions of the data set.

The advantage of using a deep ensemble rather than a Gaussian process is that it scales to a much larger number of training examples. Figure 3.4 (left column) shows five data sets generated in a Box2D environment of size 10m x 10m. To generate this data, a set of random locations with some excluded areas were chosen and random impulse forces were applied to the object at random push locations on the surface. The data is collected as a tuple of the form [*current state*, *action*, *next state*]. This data is used to train an ensemble of MDNs and are tested with test points generated within the same state-action space. The average uncertainty at a location is calculated by sampled averages performed in a 10m x 10m grid (Figure 3.4 -right column). One can observe that the ensemble approximately matches the collected data space. Blue represents low and red represents high areas of uncertainty. The parameters of the network used to learn from these different data sets are the same.

3.2.2 Model Predictive Path Integral planner

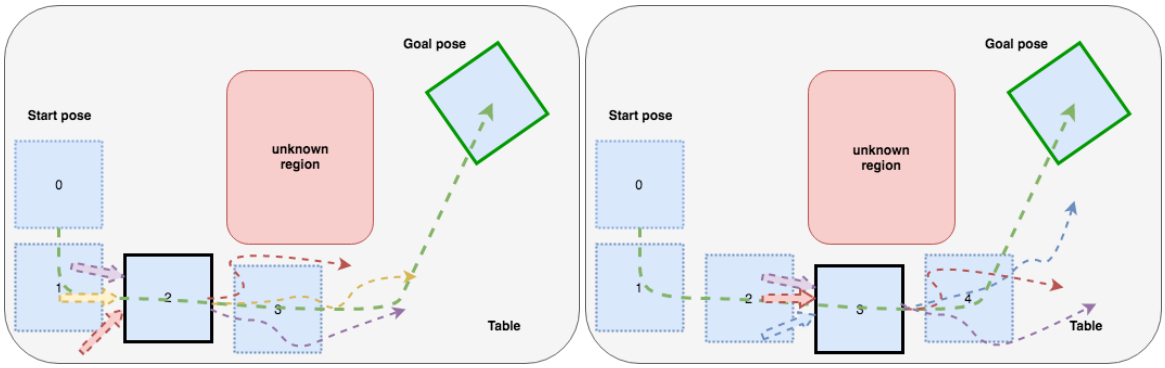


Figure 3.5: Finding of uncertainty averse trajectory from different trajectory samples. **Left:** Consider the black bordered pose as the current state (position 2) of the object. From this position, a set of random sequence (indicated as thick arrows) is applied on the object. The result of these pushes can be evaluated by simulating the forward-model. The potential trajectories generated by the model are then marked for their merit using the cost function. Based on the cost of the trajectories a push sequence is generated out of which only the first push action is executed. **Right:** Subsequently the object reached position 3. From this pose, the planner continues to generate push sequences by using random pushes and evaluating the trajectories generated by the forward-model using the cost function. The push sequence which generated the least cost trajectory (in this case, purple doted line) will have higher contribution to final control command.

The planner proposed effectively uses the forward-model to find optimal push sequence to move the object towards the goal. It is better and risk averse to exploit the known regions of the state-action space (table's surface) while moving towards the goal instead of exploring unseen parts.

We use the path integral based approach to model predictive control (Williams et al. [165]). The path integral formulation (Section 2.5.4) allows policy updates by computing cost-to-go expectations using trajectory roll-outs (Theodorou et al. [156]).Thus, instead of directly solving the non-linear Hamilton-Jacobi-Bellman equation via backward integration, the path integral formulation is used to update the commands which minimise the cost function.

The core idea of the approach is shown in Figure 3.5. Initially a random push sequence is chosen as the potential solution. Additionally another set of random push sequences are generated to approximate the gradient of the cost function (tutorial given in Section 2.5.4). Each of the push sequence is simulated through the E-MDNs based forward-model to compute the trajectory which consists of a sequence of states. The states predicted by the E-MDNs has a mean and covariance (uncertainty) associated with it. Using these values, the cost of a trajectory (and there by the corresponding push sequence) is computed using the cost-function. The trajectories which move towards the goal and lie in the known regions of the state-space receive a lower cost. These trajectory costs are used to compute the gradient of the cost function. The initial solution is updated in the direction of decreasing gradient. Thus the new updated solution will move the object towards the goal through the known regions. The mathematical details of these steps are as follows.

$$S(\tau_{i,k}) = \phi(\mathbf{x_T}) + \sum_{j=i}^{T-1} r(\mathbf{x_j}, \mathbf{u_j}, \overset{*}{\sigma}_j) \tag{3.6}$$

our cost is a function of the predictive uncertainty $\overset{*}{\sigma}_j$, in addition to the state $\mathbf{x_j} \in \mathscr{R}^n$ and controls $\mathbf{u_j} \in \mathscr{R}^m$. The exponential of the cost-to-go $S(\tau_{i,k})$ defined in Equation 3.7 provides the weight of each samples to compute their importance.

$$w_{i,k} = \frac{exp(-\frac{1}{\lambda}S(\tau_{i,k}))}{\sum_{l=1}^{K} exp(-\frac{1}{\lambda}S(\tau_{i,l}))} \tag{3.7}$$

where, λ is equivalent to the temperature parameter for the softmax distribution in Equation 3.7 and affects the control update given by Equation 3.8.

$$\Delta\mathbf{u_i} = \sum_{i=0}^{K} w_{i,k}\delta\mathbf{u_{i,k}} \tag{3.8}$$

Updates to the control command is thus calculated as a weighted average of sampled control disturbances $\delta\mathbf{u_{i,k}}$ with weights equal to $w_{i,k}$. The samples of the control disturbances

are defined as: (Williams et al. [165])

$$\delta \mathbf{u}_{\mathbf{i},\mathbf{k}} = \frac{1}{\sqrt{\rho}} \frac{\varepsilon}{\sqrt{\Delta t}} \tag{3.9}$$

with $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The parameter ρ is a scalar constant which controls the extend of the sampled disturbances $\delta \mathbf{u}_{\mathbf{i},\mathbf{k}}$. The optimal action at any state will be

$$\overset{*}{u} = \arg\min_{u}(S_i) \tag{3.10}$$

Additionally, the model predictive approach allows us to include feedback from the system. Starting at step i, a look-ahead window of T time steps is used at each state. Once the optimal control actions for these T is computed, the first command gets executed. The new achieved state after the push is fed back into the optimiser. This process is repeated till task convergence. The immediate cost function used in our formulation is

$$r(\mathbf{x_i}, \mathbf{u_i}, \overset{*}{\sigma_i}) = \gamma * \overset{*}{\sigma_i} + (\mathbf{x_i} - \mathbf{x_{goal}})^T Q (\mathbf{x_i} - \mathbf{x_{goal}}) + \mathbf{u_i}^T R \mathbf{u_i} \tag{3.11}$$

and the final cost is given by

$$\phi(\mathbf{x_T}) = (\mathbf{x_T} - \mathbf{x_{goal}})^T Q (\mathbf{x_T} - \mathbf{x_{goal}}) \tag{3.12}$$

In sample based approaches the temperature parameter λ determines the exploration of the samples. Higher value of λ allows the system to explore more and lower λ allows to refine the current solution. The choice to increase λ or decrease λ depends on the gradient of the cost function. If the cost function is getting reduced, that means the optimiser has a valid gradient and is potentially approaching a solution (local optima) and in this case it would be beneficial to decrease λ and reduce the exploration. Positive gradient of the cost function means that the existing set of samples are unable to find a proper gradient direction to minimise the cost. In this case, it is advisable to increase the number of samples by increasing the λ parameter. This update is given by Equation 3.13.

$$\lambda_p = (1 + \delta)\lambda_{p-1} \tag{3.13}$$

where, p is the iteration index and δ is a scalar in the range [0,1]. The original algorithm proposed by Williams et al. [165] is modified to incorporate the update of the temperature parameter by computing the change of the cost function. The several steps of the approach is summarised in the Algorithm 1.

Algorithm 1 : The modified MPPI algorithm, initial version proposed by Williams et al. [165]. The modifications are on (1) how the temperature parameter λ is updated to control exploration and, (2) uncertainty penalisation in the cost function.

Given:
K: Number of samples;
T: Number of timesteps;
P: Number of maximum iterations;
$(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_T)$: Initial action sequence;
Δt, time step, $\mathbf{x_0}$: Initial system state;
ϕ, r, λ: Cost parameters;
$\mathbf{u}_{init}$: Value used to initialise new controls;

while *task not completed* **do**
  for $p = 1$ **to** P **do**
    for $k = 1$ **to** K **do**
      $\mathbf{x} = \mathbf{x_0}$
      for $i = 1$ **to** $T - 1$ **do**
        $\mathbf{x}_{t+1} = f(\mathbf{x}_i, \mathbf{u}_i + \delta \mathbf{u}_{i,k})$
        $S(\tau_{i+1,k}) = S(\tau_{i,k}) + r(\mathbf{x_i}, \mathbf{u_i}, \overset{*}{\sigma_i})$
      end for
      $S(\tau_{T,k}) = \phi(\mathbf{x_T})$
    end for
    for $i = 1$ **to** T **do**
      $\mathbf{u}_i = \mathbf{u}_i + \Delta \mathbf{u}_i$ (with $\Delta \mathbf{u}_i$ given by Equation 3.8)
    end for
    compute_change_in_cost ($\Delta S(\tau_{T,k})$)
    if $\Delta S(\tau_{T,k}) < 0$ **then**
      decrease_exploration_parameter(λ, given by Equation 3.13)
    else
      increase_exploration_parameter(λ, given by Equation 3.13)
    end if
    break loop if $\Delta S(\tau_{T,k})$ no change for some iterations together
  end for
  send_control_command($\mathbf{u}_0$)
  for $i = 1$ **to** $T - 1$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$
  end for
  $\mathbf{u}_{T-1} = \mathbf{u}_{init}$
  Update current state
  Check task completion
end while

3.2.3 Finding an uncertainty averse trajectory

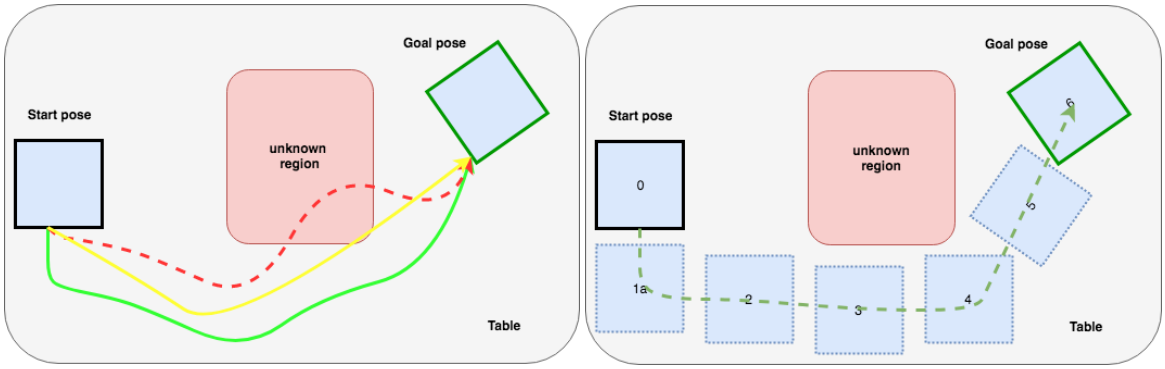


Figure 3.6: Finding of uncertainty averse trajectory from different trajectory samples. **Left:** Samples of trajectories generated. The trajectories in the known region will get least cost. In the given figure, the green trajectory will have the least cost, the red trajectory will have the highest cost and the yellow trajectory will have intermediate cost. **Right:** Next step solution. When the initial solution is updated, the most and least contribution will be by the green and red trajectories respectively. This causes the final trajectory to shift towards the location of the green trajectory.

The performance of the Algorithm 1 depends on the accuracy of the cost function used.

The challenge of the cost function defined in equation 3.6 is to optimise the trade-off between goal penalisation coefficient-Q and uncertainty penalisation coefficient-γ to help the system perform uncertainty averse push planning. Instead, we can break this problem into two optimisation problems. Firstly, learned forward-model can be used to find a lower uncertainty path and secondly the model predictive controller (Section 3.2.2) can be used to follow the trajectory generated. This way of decoupling the uncertainty cost and final goal cost shows significant improvement in the performance of the system. To solve the first challenge, we propose a path integral based approach to find a low-cost trajectory in the state-action space. The formulation derives inspiration from the STOMP planner by Kalakrishnan et al. [70] (though our formulation uses a different cost function and improvement equations). Apart from finding a low-cost trajectory, state-action space constraints are forcefully imposed on the optimisation problem to find paths within the workspace of the system.

The idea of finding an uncertain averse trajectory is shown in Figure 3.6. Imagine a flexible line (this line can be created for example by using a linear interpolation) connected between the start position and goal position of the object. This initial line can be thought as a seed solution of the path integral optimiser. Around this line, a finite number of similar trajectories are generated by adding bounded noise to the line. Each of the trajectories are penalised based on the uncertainty of the location they pass through. If few states in the

trajectory lie in a region of high uncertainty, the cost associated with these states will be high. Since the total cost of a trajectory is the sum of the cost of individual states, trajectories that pass through high uncertainty region will have higher cost.

Algorithm 2 : This algorithm optimises a trajectory with respect to model predictive uncertainty. The output of the algorithm is a trajectory with low uncertainty cost which can be tracked by a push controller, in this work we use the Model Predictive Path Integral Controller for following the pushing trajectory.

Given:
K: Number of samples;
T: Number of time steps; Number of time steps;
$f(\mathbf{x},\mathbf{u})$: Learned forward dynamics to compute uncertainty σ;
ε: Threshold of cost change;
X: $(\mathbf{x}_0,\mathbf{x}_2,...,\mathbf{x}_{T-1})$;

while *cost* $\leq \varepsilon$ **do**
  for $k = 0$ **to** K **do**
    for $i = 1$ **to** $T-1$ **do**
      $\delta X_{k,i} \sim \mathscr{T}(\mathbf{0},\mathbf{I})$
      $X_{k,i}$: $X_i + \delta X_{k,i}$
      Enforce_state_constraints($X_{k,i}$)
      $S_{k,i}$: $\overset{*}{\sigma}_i$
    end for
  end for
  for $i = 1$ **to** $T-1$ **do**
    $\mathbf{x}_i = \mathbf{x}_i + gain * \sum_{k=1}^{K} \frac{exp(-\frac{1}{\lambda}S(X_{i,k}))\delta\mathbf{x}_{i,k})}{\sum_{l=1}^{K} exp(-\frac{1}{\lambda}S(X_{i,l}))}$
    Enforce_state_constraints(x_i)
  end for
  cost = $\sum_{i=0}^{T} \overset{*}{\sigma}_i$
end while

After the cost of each trajectory is completed, we update our initial trajectory along the direction of low cost trajectories. The new trajectory can be thought as a weighted sum of all the random trajectories where the weights are determined by the inverse of the cost associated with them. The low cost trajectories will have their weights high and hence will contribute more to the final trajectory. The problem of finding a low uncertainty trajectory

can be formulated as:

$$\begin{aligned} \text{Minimise: } J &= \sum_{i=0}^{T} \overset{*}{\sigma_i} \\ \text{Subject to: } x_{min} &\leq x \leq x_{max} \end{aligned} \tag{3.14}$$

Equation 3.14 combines the uncertainty of constituent states of the trajectory. Minimising J will penalise all those trajectories lying in unknown regions. Each of the states are incrementally updated in the direction of decreasing J. However, the constraint in the equation make sure that such incremental updates on the trajectory still keep them within the spatial limits of the state-space. The trajectory found can be given to the algorithm 1 to find the right push actions. But, this time the MPPI algorithm uses Equation 3.15 with $\gamma = 0$ as the running cost.

$$r(\mathbf{x_i}, \mathbf{u_i}, \overset{*}{\sigma_i}) = (\mathbf{x_i} - \mathbf{x_{goal_t}})^T Q(\mathbf{x_i} - \mathbf{x_{goal_t}}) + \mathbf{u_i}^T R \mathbf{u_i}, \tag{3.15}$$

This Equation is a modification to Equation 3.11 with $\mathbf{x_{goal}}$ term modified to $\mathbf{x_{goal_t}}$. The immediate goals $\mathbf{x_{goal_t}}$ will be provided by the trajectory generated by algorithm 2 for each time step t. The algorithm 2 which is used to identify trajectories using Equation 3.14. The details regarding the experiment setup is described in the next section.

3.3 Hypotheses and experiments

In order to validate Algorithms 1 and 2, we conducted three experiments using the following hypotheses.

H1: Penalisation of the forward-model uncertainty helps to avoid high risk regions.

H2: The planners produce better push sequence when a forward-model learned using real data is employed.

H3: Algorithm 2 helps in faster task execution compared to using Algorithm 1 alone.

H1 tests the basic working of the approach. For Algorithm 1, **H1** means: (a) with the right combination between goal and uncertainty costs, it can generate a successful push sequence to move the object towards the goal, while avoiding uncertain regions, (b) absence of the uncertainty cost can make it generate a push sequence thereby taking the object to high risk areas. Correspondingly, evaluation of **H1** for Algorithm 2 can validate its ability to generate a less uncertain trajectory compared to the starting trajectory. Though our approach

is agnostic to the forward-model **H2** used, the same Algorithm can generate different push sequences for the same task setting depending on the model used. Finally, evaluation of **H3** would compare both algorithms each other for performance. We hypothesise that the decoupling of the cost leads to two easier optimisation problems and solving them is better than trying to solve the cost directly.

For all experiments, the E-MDNs was turned into a single Gaussian component (i.e. $K = 1$). The first experiment uses the cost function in Equation 3.11 to be optimised by Algorithm 1. The experiment shows that the right trade-off between uncertainty cost and goal cost can help the optimiser to find pushes to move an object towards the target pose while avoiding high risk regions. Figure 3.8 right and Figure 3.9 left shows the result of this experiment. They indicate that the trajectories generated when either penalised or not penalised for model uncertainty are distinct.

Evaluation of **H1** and **H3** is performed in a simulation environment. The simulation environment uses the Box-2D physics engine helps to generate large quantities of data quickly. This data is used to train the E-MDN and can easily check the merit of the approach. The simulation environment consists of a box and a point mass which can push the box in different directions. Before the start of a push, the box is kept a random location in the simulation environment. The box has a mass of 200 grams with its centre of mass coinciding with its geometric centre. Further, the box gets pushed by using a randomly sampled push action with a constant magnitude of 1 Newton. Application of each push action to the box lasts for 1 second. The integration step size of the simulation environment is 0.01 seconds. Hence, each push action lasts 1 second and contain 100 integration steps. We record a data vector consisting of the [start pose, push action, end pose] at the end of each push. Then again, we reset the box to a different location, and repeat the process till 2500 data points are collected. Next, we artificially create a void in this dataset by removing some data points. The data point removed areas act as regions of high uncertainty. These lesioned data set is used to train the E-MDN, and Figure 3.4 illustrates the result. The trained models are used as forward-models for push planning using the two algorithms proposed. For the experiment, two poses in the simulation space are chosen as start and goal pose, respectively. The push planning algorithm has to find optimal push actions to move the box from the start pose to the goal pose. The result of this experiment validates **H1**.

Testing **H2** uses a real robot. For the robot experiment, Baxter (Ret [132]) - a dual-arm robot was used. The experiment setup is as shown in Figure 3.7. The left hand of the robot contains a rod, and the right hand contains a poking peg. The robot's arms are controlled in position control mode. We use a box which weighs 250gm for the experiment. The robot

collects the data to build the forward-model by pushing the box in random directions. The 3D pose of the box is tracked with the help of *fiducial markers*(Olson [125]).

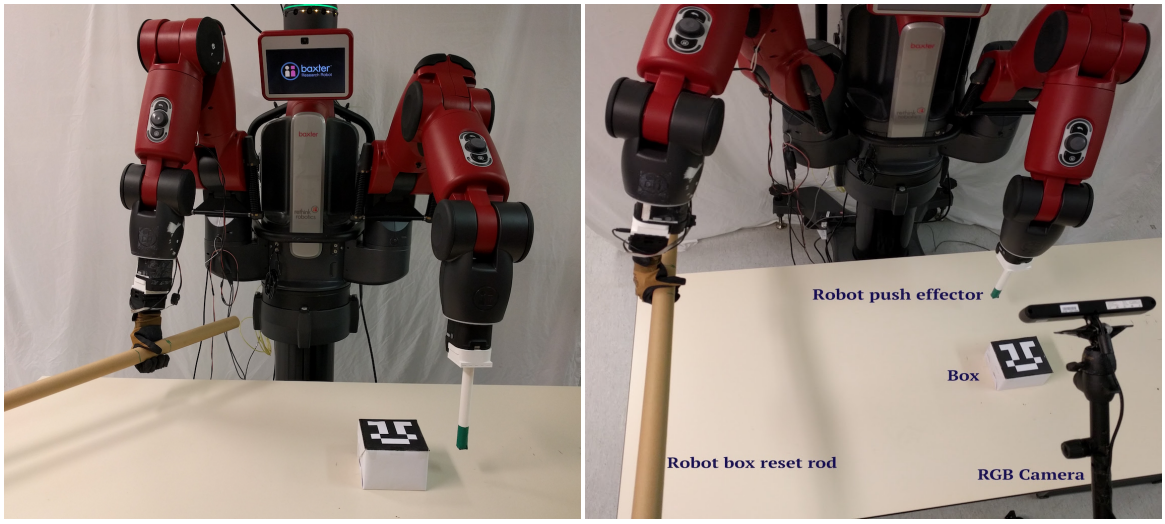


Figure 3.7: Push manipulation experiment setup for evaluating **H2**. **Left:** Push manipulation setup. The Baxter robot is setup with a self reset mechanism in the left hand to collect data from random pushes to build the forward-model. The learned forward-model is used by the planner to plan a sequence of pushes using the right hand of the robot. **Right:** Push manipulation setup with labelled parts.

At each step, we record the pose of the box, and subsequently a random push action is chosen. We uniformly sample the push directions from the range [0-1] while keeping the magnitude fixed similar to the simulation setup. There is a conversion of the sampled value to a point on the box's lateral surface. From the chosen point on the box surface, a push action is defined perpendicular to the surface at that point. After we define the push, there is a need to align the robot's poking peg to the pushing side of the box. The next step is to define a goal position for the peg. A line connecting the initial peg position and the goal position is perpendicular to the box side. An inverse-kinematics solver converts the peg's goal position to a joint state goal position. The robot moves to the target pose, thereby pushing the object. We record the final pose of the box after the push execution. Similar to the simulation setup, we collect the datum as [start pose, action value, end pose]. Here the action value is between [0-1]. From the resulting pose, another randomly sampled push action moves the box further. The robot resets the box to once it leaves the camera's field of vision. For this, the robot uses a set of predefined motions with the rigid rod attached to its right hand. Executing these predefined actions move the box back to the centre of the camera's view frame. The link here [5] shows the real experiment video footage along with the data collection procedure. After

[5]https://youtu.be/LjYruxwxkPM

collecting data, two types of forward-model are trained. The first uses the Gaussian Process (GP), and the second uses the E-MDNs. After learning the model, algorithm 1 generates a sequence of optimal push actions to move the box to a given target pose. The performance of the forward-model is compared against using a physics-engine based model. The next section shows the results of these experiments.

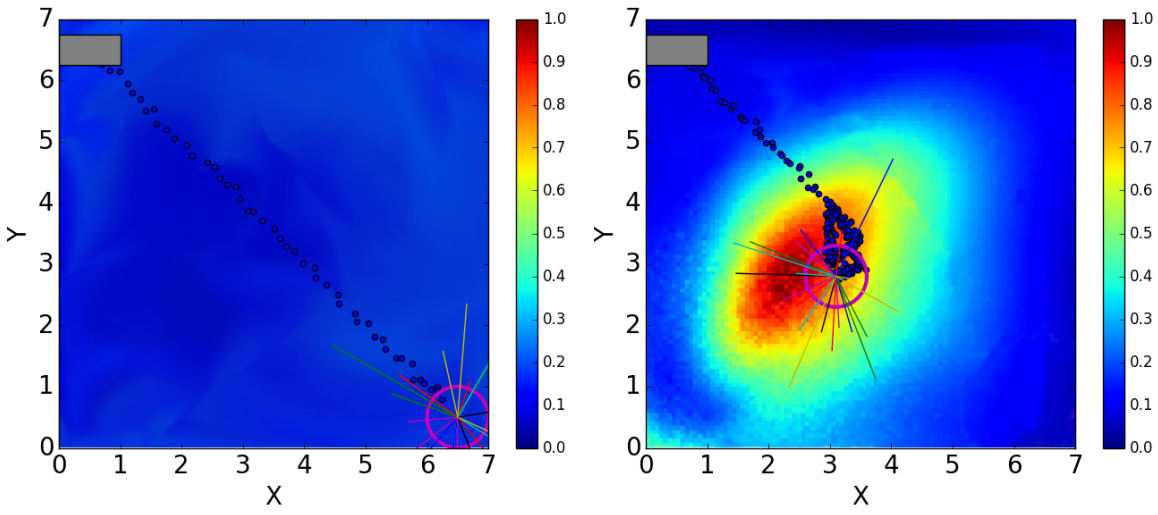
Evaluation of **H3** involves testing performance of Algorithm 2 over Algorithm 1. For this, we need to find the total time taken by each of them in moving an object from start to finish. We will use average time taken per push for a entire push sequence to measure algorithm performance. During the experiment, Algorithm 2 initially finds a trajectory with minimum uncertainty cost. Later, Algorithm 1 generates push actions required to follow this less uncertain trajectory. However, Algorithm 1 uses a modified cost function which does not penalise uncertainty when finding push actions (since the trajectory to be followed is already optimised for low uncertainty). The results for this experiment are shown in Figure 3.9 right.

3.4 Results

Figure 3.8a is to validate the entire approach for the ideal case. This figure shows that when there is no uncertainty in the state space, the box should take the shortest route to the goal. Figures 3.8b and 3.9a show the experiment's outcome after evaluating **H1**. Using the parameters specified in the figure details, we can observe that uncertainty aversion has an impact on the push sequence, thus on the object trajectory. We can see that the push trajectory of the box gets lost in the high risk region without uncertainty aversion (Figure 3.8b) while the trajectory with uncertainty aversion reaches the target (Figure 3.9a).

Similarly, result of the experiment conducted to evaluate **H3** to measure Algorithm 2 performance over Algorithm 1 shows the effect of initially finding a less uncertain trajectory (Figure 3.9b). For the given start pose and goal pose, an initial uncertainty averse trajectory was generated (doted curve in the right image). It can be seen that, the initial trajectory generated finds a better path than trying to find both trajectory and push sequence together. The push sequence generated to follow the risk averse trajectory can push the object to the goal pose. Measuring computational time of each algorithm shows that Algorithm 2 took an average of 4.34 seconds to converge per push compared to 9.78 seconds of Algorithm 1.

To learn the accurate dynamics, the subsequent experiment used a forward-model trained on real data from the robot. In order to account the inaccuracy, the forward-models chosen were Gaussian Processes (GP) and E-MDNs due to their ability to represent uncertainty. The GPs based forward-model took several steps to reach the target pose (Figure 3.10). The result shows the planner generating several incorrect pushes compared to Figures 3.11 and 3.12.



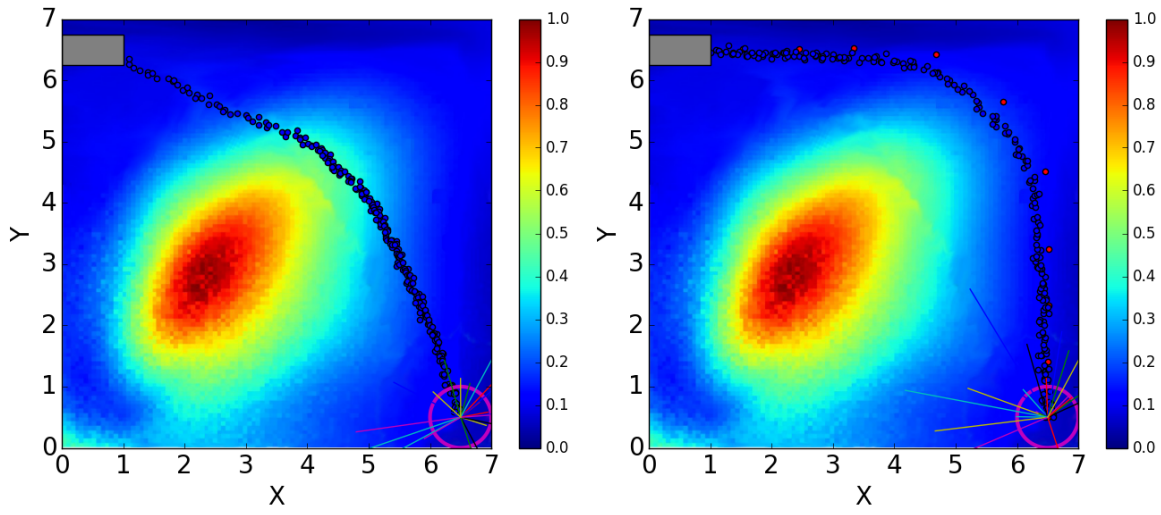
(a) (b)

Figure 3.8: The result after evaluating **H1**. The goal pose of the box is [x = 6.5, y = 0.5,$\theta = 0$]. The colour bar on the side denote the level of normalised uncertainty of the heatmap. The least uncertain regions are denoted by blue, while the most uncertain regions take red colour. **a:** Image shows when the complete data is available, the forward-model is certain about whole of the state space. When the MPPI algorithm is asked to plan a sequence of push actions to take the box to the goal pose, and when they are applied on the box, the centre of mass of the box generates the blue trajectory which is the optimal path to the goal. **b:** When there is lesion in the data, the forward-model is uncertain about certain regions of the state-space. When the box is pushed to the uncertain region, the uncertainty in the forward-model causes near random pushes on the box and gets stuck in the region.$\gamma = 115$, $R = \mathbf{I}$*0.17, $h = 2.0$, $\rho = 1.0$, $N = 3$, $\Delta t = 0.05$, $K = 20$.

This was mainly due to the inability of the method to scale with the size of the data set. Using of partial number of data points to train the model resulted in an incomplete forward-model.

Next, the same data was used to train an E-MDN model. Algorithm 1 was used to find a push sequence to move the object to the goal. The planner was able to take the box to the goal in least number of steps (Figure 3.11).

The planner took more number of steps to reach the goal pose when the physics engine was used (Box-2D) as the forward-model (Figure 3.12). This is due to the inaccurate approximation of object dynamics on the table surface. We can observe that the GP also took more number of steps compared to the E-MDNs. One reason for this is the ability to E-MDNs to predict the resulting box orientation more accurately than that of the GP. Incorrect predictions of orientation can cause several incorrect pushes thereby taking more number of steps. However, it can be seen that GP was also able to capture better dynamics and reach the target in less number of steps. Thus, this experiment proves **H2** and concludes



(a) (b)

Figure 3.9: The result after evaluating **H3**. The goal pose of the box is (x=6.5, y=0.5,$\theta = 0$). The blue regions denote the least uncertain regions (normalised uncertainty of the heatmap) while the red denotes the most uncertain regions. **a:** Performance of Algorithm 1. Penalising uncertainty, the box is deflected towards regions with lower uncertainty. The uncertainty penalty only acts for the first 150 pushes and then made zero ($\gamma = 0$). For this experiment gamma was $\gamma = 115$, $R = \mathbf{I}0.17$, $h = 2.0$, $\rho = 2.0$, $N = 3$, $\Delta t = 0.05$, $K = 20$. **b:** Performance of Algorithm 2. The red dotted trajectory is the way-points generated by just penalising uncertainty in the first stage of the algorithm. These way-points are given to the MPPI algorithm with the modified cost function to generate a set of pushes. It can be seen that the initial trajectory generated is better than the one produced by Algorithm 1. For this experiment gamma was $\gamma = 115$, $R = \mathbf{I}$*0.17, $h = 2.0$, $\rho = 1.0$, $N = 3$, $\Delta t = 0.05$, $K = 20$.

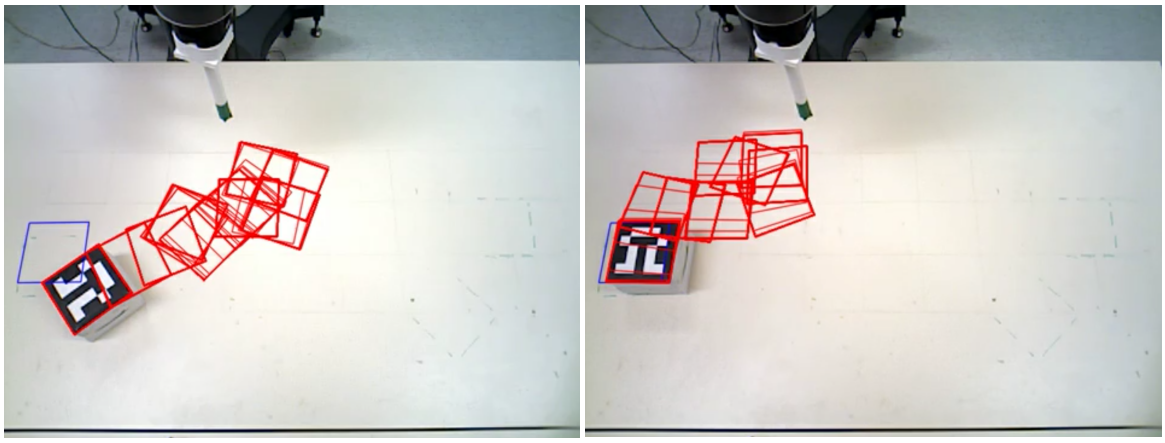


Figure 3.10: Pushing using Algorithm 1 with Gaussian Process Regression as the forward-model for the evaluation of **H2**. The task is to push the box to same target location from two different start poses. **Left:** Start pose 1 **Right:** Start pose 2

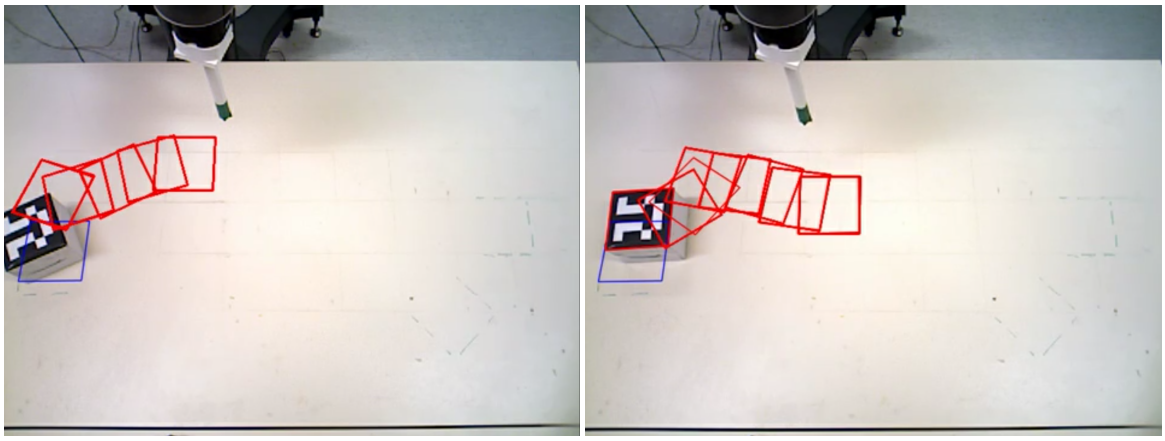


Figure 3.11: Pushing result of the Algorithm 1 using Mixture Density Networks as the forward-model for evaluating **H2**. The task is to push the box to same target location from two different start poses. **Left:** Start pose 1 **Right:** Start pose 2

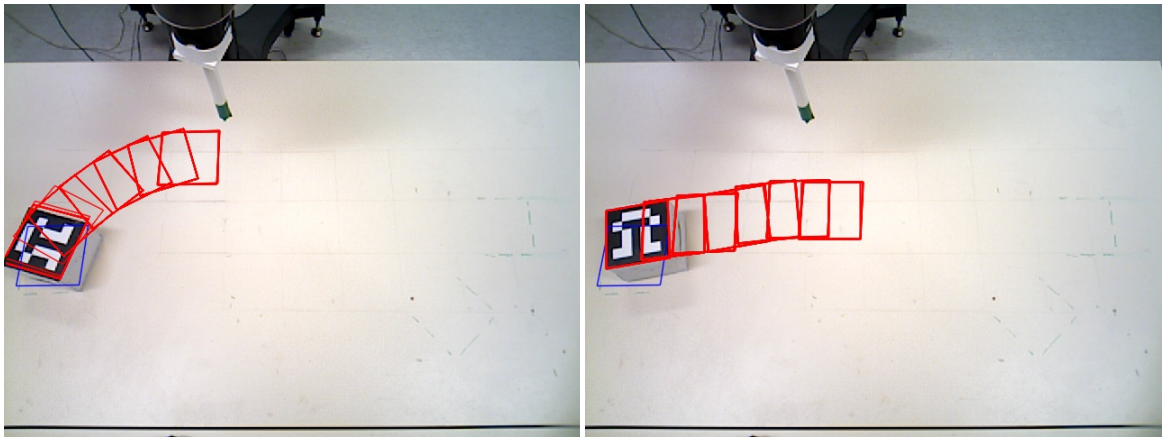


Figure 3.12: The baseline case of using Algorithm 1 with physics engine as the forward-model for the evaluation of **H2**. The task is to push the box to same target location from two different start poses. **Left:** Start pose 1 **Right:** Start pose 2

that real data-based models work better than physics engine models for performing push manipulation.

3.5 Discussion

Planning robot actions to push manipulate objects require reliable learned models. This chapter proposes two approaches which combine the idea of forward-models with a model predictive controller. The approaches can plan and execute push sequences to move an object towards a goal. The planner is agnostic to the type of forward-model used. Experiments use

different forward-models such as learned models and physics engine based. The approach generates push sequences which move the object towards the goal through the known parts of the state space. The known and unknown regions of the state-space can be differentiated using the predictive uncertainty of the learned model (E-MDNs, GP).

Initially, the algorithm was tested on a toy data set generated using the Box-2D physics engine. Generating any amount of data from physics engine takes only a fraction of time compared to real data and hence can be used for quick testing of the algorithms. The collected data was lesioned artificially to remove some data points. These lesions acted as regions of high risk. Subsequently, the forward-model trained with this data produced high uncertainty in the lesioned area compared to other regions. This trained model was used by both algorithms to find push commands. The training of the models required a lot of parameter tuning. There were many instances of model convergence to local minima leading to improper representation of the data. In our model, the neural network outputs parameters of Gaussian distribution. We observed that it is better to start by learning parameters of a single kernel and subsequently increase the number of kernels as required by the model complexity. It is important to find a good seed for the random number generator which is used to initialise the neural network model. The early convergence or local convergence of the optimiser can happen depending on this initialisation. Later these models used in the model predictive control framework involved another set of hyper-parameter tuning. These parameters relate to the number of sample trajectories required to estimate the push, the number of look ahead steps required in each trajectory, the exploratory noise parameter, decay of the noise parameter and weighting of individual terms of the cost function. The parameter we chose are depicted in earlier sections. Selection of these parameters are more of an art than science. The need for us to introduce a decay parameter for the exploration was to force the optimiser converge to a good enough solution. Without use of the decay parameter results in slow convergence and thereby taking longer time in reaching the goal pose.

All the forward-models were successful in completing the task though each of them used different number of pushes. It can be seen that the physics engine based experiment took the most number of steps. This is because the physics engine failed to capture the correct dynamics of the object. The inaccuracy in the object's dynamics leads the planner to take more number of incorrect pushes. In Algorithm 1, the MPPI uses the cost function containing a direct penalisation for the model uncertainty. This caused the algorithm to take more number of iterations to optimise (since the optimiser had to satisfy two constraints simultaneously). The optimiser had to find push actions that move the object towards the goal through less risky regions. Instead of solving two costs like in the MPPI planner, the

second algorithm decouples the costs into two. Initially, algorithm 2 is used to generate an uncertainty averse trajectory. In the subsequent stage, the MPPI planner with modified cost function (Equation 3.15) finds the pushes required to follow the trajectory generated. This approach made the MPPI algorithm to run faster in less number of iterations. The real robot experiments produced best results when E-MDNs were used as the forward-model. The learned model trained on real data took least number of pushes to complete the task owing to the accuracy of the forward-model.

Algorithm 1 finds an optimal push by balance between the uncertainty penalisation and goal penalisation. The challenge is to come with right penalisation coefficients and the tuning may take some time. Algorithm 2 does not require this careful balance of these coefficients. However, this results in the challenge of knowing when to stop the algorithm to decide how much uncertainty will not hurt. It should be also noted that the forward-models used were not tested on data sets having multiple lesions. Though the E-MDNs proposed can be used to represent such situations, it would be a worthwhile experience to try out their performance. Another issue to be noted is that, both the planners assumed lack of obstacles in the state space. If this assumption is relaxed, there is a chance of algorithms getting stuck in an obstacle unless obstacle specific term is inserted to the cost function to avoid them.

Chapter 4

State dependent stiffness control

This chapter describes a part of the second thesis contribution, which was presented in Mathew et al. [108] and Mathew et al. [107]. Here, we aim to develop a framework which performs anticipatory control and variable impedance in continuous interaction robot tasks. This contribution consists of two parts. The first part (presented in this chapter) details the proposed architecture and the second part in Chapter 5 describes the experiments conducted to study and validate the framework.

The chapter is structured as follows. We start with our motivation and problem statement (Section 4.1) followed by the details of the proposed framework (Section 4.2). The chapter ends with the discussion (Section 4.3) describing an overview of the approach.

4.1 Motivation and problem statement

Manipulation tasks such as polishing, assembling complicated parts with tight tolerance, deburring, dexterous in-hand manipulations involve complex contact interactions with the environment. These tasks require following desired trajectories while maintaining necessary contact forces. For example, polishing a table surface will require the application of a downward force to keep the scrubber on it. Opposing forces also get applied against the surface friction to enable smooth motion (while following the specified trajectory). Hence, reasonable strategies are required to control motion and force when there is contact between robot parts and the surface of objects. Control of such tasks requires the knowledge of the dynamics of the robot and environment under interaction.

Traditional robots working in closed cages have the liberty to be arbitrarily stiff since they work in a structured environment (i.e. specially designed environments) where the changes of unpredictability are almost nil. Pre-programing each interaction with the environment using traditional techniques become cumbersome as robots are getting used in unstructured

environments and possibly collaborate with humans. The challenge arises due to the difficulty in obtaining accurate values of inertia, frictional and damping parameters of the interacting environment. Plans to use robots in less structured environments demand a radical shift from traditional control strategies. However, most of these strategies are insufficient to facilitate a certain level of compliance (without compromising task accuracy), which is required to deal with the unstructured environments.

Planning of trajectories in free space is well studied and can be achieved using rigid body dynamics. In principle, a motion planner can plan and execute an entire interaction task accurately, if the complete interaction model of the robot and environment is available. However, it is tough to obtain the dynamic model of the environment, though the robot model is accessible with sufficient accuracy. Modelling error results in planning error and subsequently causes deviation from the desired trajectory due to incorrect estimation of contact forces. Accumulation of such errors leads to reaching of joint limits, application of torques above safety limit at the end effector, high impact forces at interaction points, thereby resulting in the damage of the robot. Also, the contact interactions are highly non-linear, making the generated plans unusable. Performing contact manipulations demands reasoning about the forces and torques required to achieve the task goals. Mostly, measuring contact forces can describe environment properties. However, safe collection of the required environment information (Duan et al. [42]) necessitates a compliant behaviour of the robot.

Usually, force measurement is achieved using a force-torque sensor mounted between the wrist and the end-effector. Traditionally, controlling force in known environments are performed using impedance control and hybrid force-position control schemes. The right set of impedance parameters of the end-effector helps in controlling the force of resistance due to external motions imposed by the environment. However, when the environment varies dynamically (e.g. a surface to be polished with several abrupt notches), it becomes difficult to model the environment beforehand and hence tracking becomes a challenge. Human beings perform such complex tasks with ease. This dexterity is primarily due to the ability of the human brain to predict environment interactions and vary the impedance of the interactions. Similarly, prediction of environment dynamics (interaction forces) resulting in better control. Now we start with formally stating the problem.

Consider two tasks: polishing a table, and stirring a porridge. Polishing the table involves following a particular trajectory on the surface of the table while applying a downward force. However, frictional forces oppose the motion on the table surface. For the task control, the robot will have to maintain a specific set force on the surface while following the desired trajectory. Similarly, stirring the porridge involves following a particular trajectory in a dynamically changing environment. The porridge - a non-Newtonian fluid - when stirred by

the robot will experience an increasing opposing force as the task progresses, owing to the property of such fluid to change its viscosity on stirring. In both cases, depending on the task requirement, the robot will have to track the desired trajectory without getting hindered by the external forces.

There is a difference between them: The first task, defined in a static environment, has a need to control force and motion independently. That would mean that the controller has to move on the surface, compensating the frictional force while applying the required force on the table surface. In the second task, the porridge thickens which leads to an increase of the opposing forces along the stirring direction. The environment is dynamic, and the robot has to control force and motion in the same direction. The interesting problem is whether we can design a single framework to solve both the tasks. Since estimation of environment parameters is difficult, we can think of directly learning the interaction dynamics from data.

4.2 Proposed solution

Humans can outperform robots in their dexterity even with their neural pathway delays (80ms Mitrovic et al. [116]). They can account for the delays by learning and thus creating internal models of the tasks. These internal models help them to anticipate interaction dynamics and thereby equip their hands to act better while task execution. Traditional control schemes mostly work on negative feedback architectures and hence, the control action happens only once the end-effector has felt the force. It would be better to act or anticipate control and act before sensing the force. This way of control is possible if the framework can anticipate forces of the next state from the current step.

Research indicates that humans performing tasks like porridge stirring or board polishing, which involve continuous interaction with objects or the environment, typically use higher (arm) stiffness when they perform the task for the first time. The higher stiffness in the initial trials help to improve task accuracy and counter any unforeseen disturbances. With sufficient experience, humans learn internal models of the task's dynamics and use these models to predict the *task state*, i.e., the configuration of the object(s) and hand in the task context. The task state dictates the variation in stiffness, and the choice of these parameters significantly influences performance (Kawato [74]). Studies in psychophysics also indicate that learning to vary stiffness is crucial in manipulation tasks (Franklin et al. [52], Burdet et al. [21]). Humans with an excellent learned model can adapt stiffness to the task, typically performing the task with much less stiffness than before.

Task-space controllers (Cartesian-space controllers) are intuitive and have task-specific parameters. Hence, usage of the framework becomes independent of the type of robot

manipulator, and the reuse of the learned model is possible in other robots. The framework also should be able to predict the forces due to the dynamics of interactions. For predicting the forces, the robot has to learn the forward-model of the interactions. A forward-model estimates the forces the robot might experience in the next instance and hence, an appropriate feed-forward value can be used to compensate for the forces. For example, in the porridge stirring task, the prediction of forces will allow the robot to stir the liquid along the desired trajectory without being hindered by the changes in external force.

The motivation for the proposed framework (Section 4.2.2) is backed by evidence from human motor control literature described in Section 4.2.1. Details on how variable impedance (Section 4.2.3) is defined as a function of the forward-model and the specification of compliant directions is presented in Section 4.2.4.

4.2.1 Evidence from human motor control studies

The biological motor systems exhibit impressive performance although they are affected by significant noise, sensory delays and other stochasticity (Faisal et al. [45]). Human beings have the capability to adapt impedance parameters of the overall biomechanical system during task execution to account for these stochastic disturbances (Buchli et al. [19]). This ability results in versatility and robustness of human motor control system. A similar behaviour in robot manipulators enable them to work safely in human environments. But adopting these techniques for the control of high degree of freedom manipulators is non-trivial. Classical robot control schemes are based on high gain negative error feedback control while biological systems use low gain compliant control with task dependent variable impedance (Selen et al. [143]). Approaches for adaptive impedance control were attempted through time varying proportional and derivative gains, also known as gain scheduling (Lin and Lin [98]). However, it is hard to find appropriate gain schedule for a given task due to inaccuracies in model estimation and sensor measurements (Hogan [62], Siciliano et al. [147]).

It is observed that the human motor system uses predictive models of the effects that motor actions have on sensory data (Flanagan et al. [51], Johansson and Cole [68]). These predictions are used for different purposes such as feed-forward control, motor system coordination, action planning and monitoring. Inspired by human control literature such as Kawato [74], Shadmehr and Krakauer [145], equipping a robot manipulator with the ability to learn a predictive forward-model of the task environment will enable the robot to predict the forces it may experience in the next state, and apply appropriate control signals to counter these forces. The predictions provided by such a forward-model can then be used as a feed-forward term in the control command. Incorporating a feed-forward term along

with the feedback term in the controller has shown to reduce the number of training samples required for learning a variable impedance policy (Alberto et al. [8]).

4.2.2 Framework

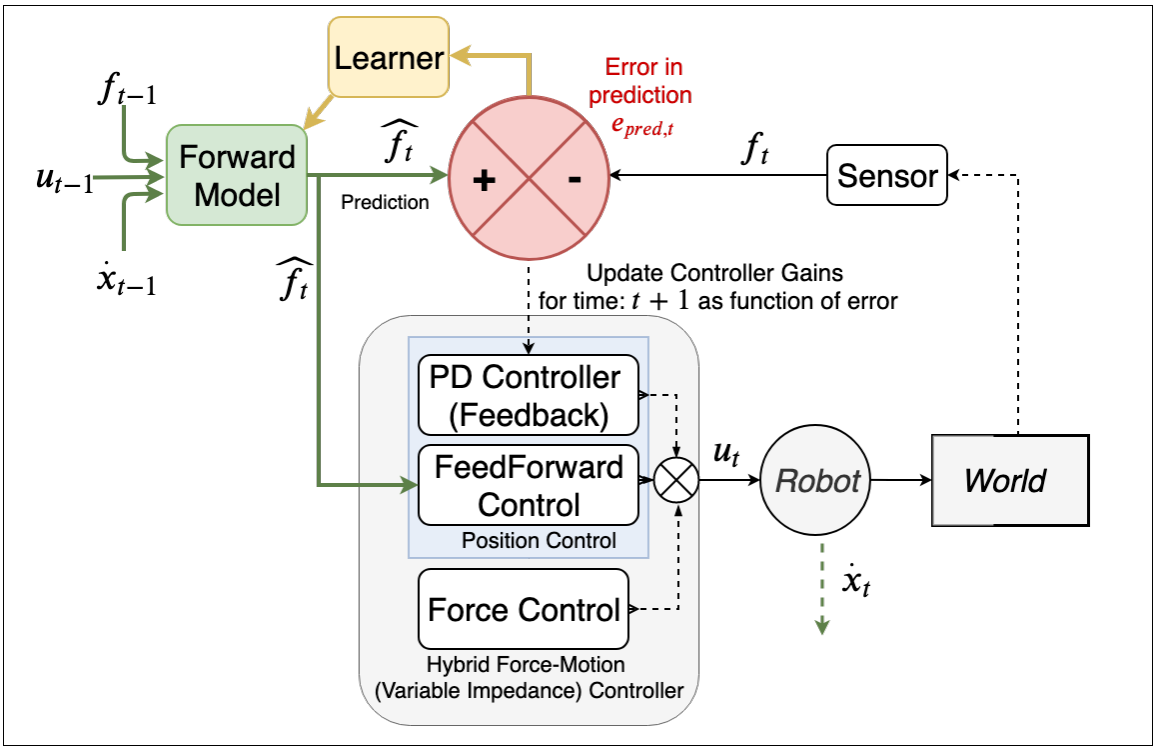


Figure 4.1: Framework

The overall block diagram of the framework is as shown in Figure 4.1. For tasks which involve control of force and motion along same directions, the robot must learn to adjust the stiffness to apply varying forces. The force required at each instant is dictated by the task configuration.

The impedance control is developed and deployed in the task-space of the robot. Task-space controllers are typically designed as a *mass-spring-damper* system. The goal of the controller is to make the robot behave like a spring attached between the end-effector tip and the motion *waypoint* [1]. Shaping inertia of the robot to behave like a mass-spring-damper system is generally tricky (Wimbock et al. [166]) and requires accurate measurement of the external forces acting on the robot. Since measuring forces on every acting point is impractical, it is difficult to modulate the inertia tensor without leading to incorrect impedance behaviour. For this reason, in practice, the desired impedance behaviour is limited to

[1] A waypoint is an intermediate point on a trajectory or a point at which the motion profile changes.

designing stiffness and damping parameters of the controller while keeping the natural inertia unchanged, resulting in the compliance control problem (Wimbock et al. [166]). However, arbitrarily varying stiffness and damping parameters may result in unstable behaviour of the robot (Kronander and Billard [81]). In all our experiments, we use empirically found bounds for stiffness and damping parameters (upper bound: 1000 N/m, lower bound:100 N/m).

Equipping a robot manipulator with the ability to learn a predictive forward-model of the task environment will enable the robot to predict the forces it may experience in the next state and apply appropriate control signals to counter these forces. The predictions provided by such a forward-model can then be used as a feed-forward term in the controller command. The corresponding controller equation can be written as:

$$\mathbf{u_t} = \mathbf{K_t^p}\Delta\mathbf{x_t} + \mathbf{K_t^d}\Delta\dot{\mathbf{x}}_\mathbf{t} + \mathbf{k_t} \tag{4.1}$$

where $\mathbf{u_t}$ is the control command to the robot at time t, $\mathbf{K_t^p}$ and $\mathbf{K_t^p}$ represent the (positive definite) stiffness and damping matrices of the feedback controller respectively; $\mathbf{k_t}$ is the feed-forward term provided by the forward-model; $\Delta\mathbf{x}$ and $\Delta\dot{\mathbf{x}}$ are the errors in the end-effector position and velocity.

During the training phase, the forward-model gets initialised by executing the provided demonstration in high stiffness. This forward-model is further improved online in subsequent trials by the robot. At each state ($\mathbf{x}$), the system is trained in a supervised manner to predict the next state force given the current state.

During task execution, the predictions from the forward-model are used as a feed-forward term ($\mathbf{k_t}$) in the controller, while the error in the prediction controls the feedback gains ($\mathbf{K_t^p}$ and $\mathbf{K_t^p}$). Revision of the feedback control gains happens in proportion to the prediction error. As the model is updated online, the error in predictions is expected to reduce over time, and the feedback gains can be adjusted appropriately.

Figure 4.1 is a graphical representation of the proposed framework. The feed-forward component of the controller uses the prediction from the forward-model. This, together with the feedback component, provides the motion control command to the robot. The difference between the measured and predicted state, i.e., the error signal, is used to update the model and to decide the subsequent feedback gains for the motion controller. The force control is performed in the directions orthogonal to motion when required. This way of separating force and motion in the task-space can be used to separate the directions to be stiff and compliant.

If the robot has a forward-model to tell what forces it may experience in the next instance, an appropriate feed-forward value can be used to compensate for the forces. This will allow the robot to execute a task without being hindered by the changes in external force.

The proposed framework uses a Gaussian Mixture Model (GMM) to represent the forward-model. The GMM is fit over points of the form $X_t = [S_{t-1}, f_t]$, where S_t can be any combination of features that can uniquely represent the state of the robot for the task at an instant, and f_t is the force felt at the end-effector at time t. The state vector S_t can contain information in terms of end-effector position (x_{t-1}), velocity ($\dot{x}_{t-1}$), forces (f_{t-1}), etc. The model is fit using the Expectation-Maximisation (EM) algorithm, where the following likelihood function is maximised:

$$L(\theta) = p(\mathbf{X}|\theta) = \prod_{n=1}^{T} p(X_n|\theta) = \prod_{n=1}^{T} \left[\sum_{j=1}^{M} p(X_n|j)p(j) \right]$$

where $\theta = (\mu_j, \sigma_j, p_j)$ for $j = 1...M$ are the parameters of the M components of the mixture model. $\mathbf{X} = (X_1, X_2, ..., X_T)$ represents the points to be fit by the GMM, with $X_t = [S_{t-1}, f_t]$.

Note that each point contains the information about the *previous* end-effector state, along with the *current* force. Hence, when the model is used for prediction during task execution, the force for the next time instant can be predicted using the current state of the robot. The force at next instant $(f_{t+1}|S_t)$ is computed from the learned model using Gaussian Mixture Regression (GMR) (Sung [154]).

A variant of GMM called Incremental GMM (IGMM) (Song and Wang [148], Ahmad [5], Engel and Heinen [44]) is available which can incrementally learn a mixture model for the joint density of the incoming data. IGMM incrementally updates the density estimate taking only the newly arrived data and the previously estimated density. It is also able to create a new mixture component if the incoming datum is below the acceptable likelihood for the current model.

The proposed framework makes use of IGMM to learn and improve the forward-model *online* during task execution. Due to the fast incremental learning, the approach can quickly adapt to different tasks when initialised with a model learned for a similar task. This is empirically validated using the board polishing task (by a robot) described in Section 5.3.

4.2.3 Varying feedback gains

Consider a robot pulling a spring in different directions, it is easy to see that the robot can perform the task successfully with very high stiffness ($\mathbf{K}^p_{max}$), but this would require the robot to become less compliant throughout the task, hence spending more energy. If the robot had to perform the same pulling action, but this time in free-space (without the spring), it would require a much lower stiffness value $\mathbf{K}^p_{free}$) for accurate trajectory tracking. If the learned

forward-model is accurate, then the feed-forward term should cancel out the external forces in the task, essentially making it motion in free-space. Hence, the more accurate the learned model becomes, the lower the feedback gains can be (closer to $\mathbf{K}^p_{free}$), similar to human behaviour for a new manipulation task.

The feedback gains at each instant ($\mathbf{K}^p_t$) in the controller (Equation 4.1) can therefore be varied according to

$$\mathbf{K}^p_t = \mathbf{K}^p_{free} + F(e_{pred,t-1}) \times (\mathbf{K}^p_{max} - \mathbf{K}^p_{free}) \tag{4.2}$$

where $e_{pred,t}$ is the error in prediction of the forward-model at time t, and $F(x)$ is a function that maps $x \rightarrow [0,1]$, such as the logistic function. This way of modulating the feedback term ensures that if the model is predicting accurately, the robot will be more compliant, while wrong predictions ensure that the robot becomes stiff for following the trajectory more accurately. Since the forward-model is updated online, the error in predictions is expected to reduce over time, and the feedback gains should become lower.

The damping term is updated using the constraint of the damping factor for an critically-damped system (Ijspeert et al. [67]), given by the following equation:

$$\mathbf{K}^d_t = \sqrt{\frac{\mathbf{K}^p_t}{4}} \tag{4.3}$$

This equation makes a fixed ratio dependence between the stiffness and damping parameter thereby making the learning problem smaller. From some of our past experiments (Appendix C), we understood that the learning of stiffness and damping as independent parameters is hard due to the large search space. Instead, we can use the knowledge from classical control theory to make the parameters dependant.

4.2.4 Specifying the directions of compliance

Not all manipulation tasks can be achieved using the above formulation. Some tasks require the robot to become compliant when it experiences unexpected force. Consider a task where the robot has to polish a planar surface. Here, the task requires the robot to follow the trajectory defined in the plane of the surface while maintaining the target contact force in the orthogonal direction. Suppose the surface is raised suddenly during the task; then the model used in the above formulation will predict incorrect forces leading the robot to increase its stiffness. This would result in the robot trying to push the surface down, damaging itself and/or the surface, which is not the desired behaviour.

On the other hand, if the robot experiences unexpected forces along the directions (X,Y) of the task plane (e.g. frictional forces), the robot has to become stiffer to follow the trajectory more accurately and incrementally learn the forward model. Therefore, unexpected forces along different directions demand different behaviour from the robot – some directions require the robot to become stiff in the presence of unexpected forces, while others require compliance.

The most intuitive and straightforward way of separating the 'compliant' and 'stiff' directions is using a hybrid force-motion control framework. This controller works by defining *artificial* constraints to the robot's degree of freedom. The task expressed in terms of these constraints which specify desired values (to be imposed by the control law) of velocities in the k directions feasible for motion, and forces in the remaining $(6-k)$ directions feasible for contact reaction.

By using direct force control along the directions to be compliant, and motion control for the directions to be stiff, the robot can maintain the required force in the normal direction while following the trajectory on the surface. These directions can be defined manually for each task. The final control equation thus becomes:

$$u_t = \mathbf{K_t^p}\Delta x_t + \mathbf{K_t^d}\Delta \dot{x}_t + k_t + u_{force\_control} \tag{4.4}$$

where $u_{force\_control}$ specifies the command part produced by the direct force controller so as to maintain the desired force along the specified direction(s).

4.3 Discussion

In humans, the impedance regulation relies on muscle activation and neural feedback. This impedance regulation helps in learning the forward-model of a task. Probably they use a combination of these internal models and impedance adaptation to perform intricate manipulation tasks. This chapter proposes a novel framework for learning forward-model of interaction tasks and also use the error in model predictions to change the impedance parameters of the task. Motor control literature provides strong evidence in support of the proposed framework. The proposed framework does not possess any tunable parameters. To validate this framework, we will need to formulate hypotheses and conduct experiments.

For the forward-model proposed in Chapter 3, the input to the neural network was the current state and action, and the output was the parameters of a Gaussian mixture model. This formulation is equivalent to having a joint distribution of the input and output vector as

a Gaussian mixture model (GMM) and then later performing a condition operation on this GMM (using the input vector) to generate another GMM about the output vector. We can see that the forward-model used in this work is different from the one used in Chapter 3. The major reason for not using the same method is the requirement of online-learning and the bitter experience of hyperparameter tuning. Compared to the forward-model in Chapter 3, the number of kernels in the proposed method are not fixed and are generated based on a novelty criterion proposed in Engel and Heinen [44]. Fundamentally a new kernel gets generated if the existing GMM cannot explain the new incoming datum. Existing GMM cannot explain the incoming datum if its likelihood is not within a manually defined threshold. That is, a high threshold will lead to more kernel generation, and a low threshold leads to having less number of kernels. The number of kernels has an impact on the learned model. More the kernels finer will be the model. However, this comes with a computational cost. The number of operations required for the inversion of the covariance matrices is cubic, and this grows as the number of components increase. Hence, the threshold of the novelty criterion should be defined carefully.

The next crucial point is the noise of the force-torque sensors. These sensors are quite noisy, and hence their readings can cause trouble to the learner. Since the novelty criterion determines the number of kernels, noise in the sensor readings can misrepresent as false data points and cause the generation of unwanted kernels. Though there is an automatic procedure to delete unwanted kernels, our experiments suggest using raw data results in unwanted kernel generation and subsequent wastage of time for a prediction. A simple way to avoid the noise is to use a low pass filter. In our framework, we use a third-order low pass filter implemented as a zero-phase Butterworth filter.

Further, we propose a constraint between the proportional and derivative gains through equation 4.3. This constraint means that the derivative gains are dependant on the proportional gains. Ideally it is good to vary proportional and derivative gains independently. However, independent modification of both gains requires precise knowledge of the dynamics of the system (robot and interaction dynamics). We have noticed that independent variation of the gains leads to unstable robot behaviour. The ideal values of the gains are dependant on system dynamics.Since we do not have a precise model and we learn the dynamics online, it is better to vary the gains dependently to avoid unstable behaviour. Keeping this constraint allows the adapter values to be critically damped and hence avoid oscillation of the system. However, this also results in a temporal lag in the performance of the system. This is the reason for the temporal lag observed in the experimental results presented in the next chapter.

To establish the capability of the framework, we choose both static and dynamic environments to perform different tasks. We conduct a non-linear spring pulling (Section 5.2)

experiment to verify the framework behaviour in dynamic environments. This capability is further examined by using a rapidly varying porridge stirring experiment (Section 5.4). Next, we explore the generalisability of the framework to novel environments. To test it, we use a board polishing task (Section 5.3) defined in a static environment.

Chapter 5

Experiments using the framework

In this chapter we conduct different experiments to validate the framework proposed in Chapter 4. This framework can learn the forward-model of a task to enable the robot in executing it at a lower stiffness without affecting the overall task performance. The model learned is capable of predicting the next state force, thereby helping the controller module to apply control in an anticipatory fashion. At each controller step, the prediction of the forward-model gets compared with the feedback force sensor measurement. Depending on the error in the prediction, the controller decides whether to change the impedance parameters. This behaviour is similar to human motor control (Kawato [74], Flanagan et al. [51], Rosenbaum [134]), where the hand is stiffened to improve task accuracy when the predictions get inaccurate.

We start by stating the hypotheses (Section 5.1) used to validate the framework followed by details of three experiments. Next, we specify the network architecture of computers used in the experiments and the reason for adopting it. The first experiment is described in Section 5.2 followed by second experiment (Section 5.3) and third experiment (Section 5.4).

5.1 Hypotheses and experiments

We conducted three main experiments to evaluate the following hypotheses about the capability of our framework.

H1: Using feed-forward-model along with stiffness adaption improves trajectory tracking performance.

H2: Adding efferent copy as input to the forward-model creates a better model of task dynamics, resulting in improved trajectory tracking.

H3: Online learning of the forward-model supports adaptation to new and changing environments.

H1 tests the effectiveness of using the forward-model and subsequently adapting stiffness based on the accuracy of the model; **H2** compares the choices of the feature vector (4.2.2); and **H3** assesses if the framework can adapt to new environments. Root Mean Square(RMS) measure is used to quantify the error in achieving the desired motion profile, along with suitable plots which provide a qualitative indication of performance. We do not provide an experimental comparison since our approach is significantly different from popular approaches for such manipulation tasks (e.g.,based on deep learning). The obvious advantages of our approach are discussed in the 'Result' section of each experiments. For testing **H1** and **H2**, the experiments are compared in four different stages. Performing the task in the:

1 absence of a forward-model under high constant stiffness.

2 absence of a forward-model under low constant stiffness.

3 presence of a forward-model under low constant stiffness.

4 presence of the forward-model with varying stiffness.

The framework consists of a state-dependent stiffness controller. Different types of feature vectors are chosen to represent the state. The choice of features was motivated by various computational motor control literature such as Shadmehr and Krakauer [145]. In our experiments, the forward-model is learned online as described in Section 4.2.2, with the feature vector $p = [S_{t-1}, f_t]]$. For testing **H1** and **H3**, the state vector was chosen to be $S_t = [\dot{x}_t, f_t]$ whereas in order to test **H2**, the forward-model uses the computed force vector in the task space (u_t) in addition with the existing feature vector; $S_t = [\dot{x}_t, f_t, u_t]$. The forward-model IGMM learns the parameters which models the probability distribution of the feature vectors. Later this learned model can be used to condition on S_t to predict f_{t+1} using Gaussian Mixture Regression (GMR).

To test the hypotheses, four sets of experiments were designed using a 7-DoF Sawyer robot [1]. The tasks are executed by making use of a Cartesian space hybrid force-position controller. The directions in which force and motion is controlled are task specific. Hence, the directions are to be mentioned in advance as a hyper-parameter. The framework predictions are only used in the directions of motion control. Note that, we empirically found the bounds within which the stiffness and damping parameters are stable for the Sawyer robot. Since varying these parameters beyond a certain range may result in unstable behaviour of the robot (Kronander and Billard [81]).

[1] Experiment Video: https://youtu.be/vjJwexVziS0

5.2 Pulling linear and non-linear springs

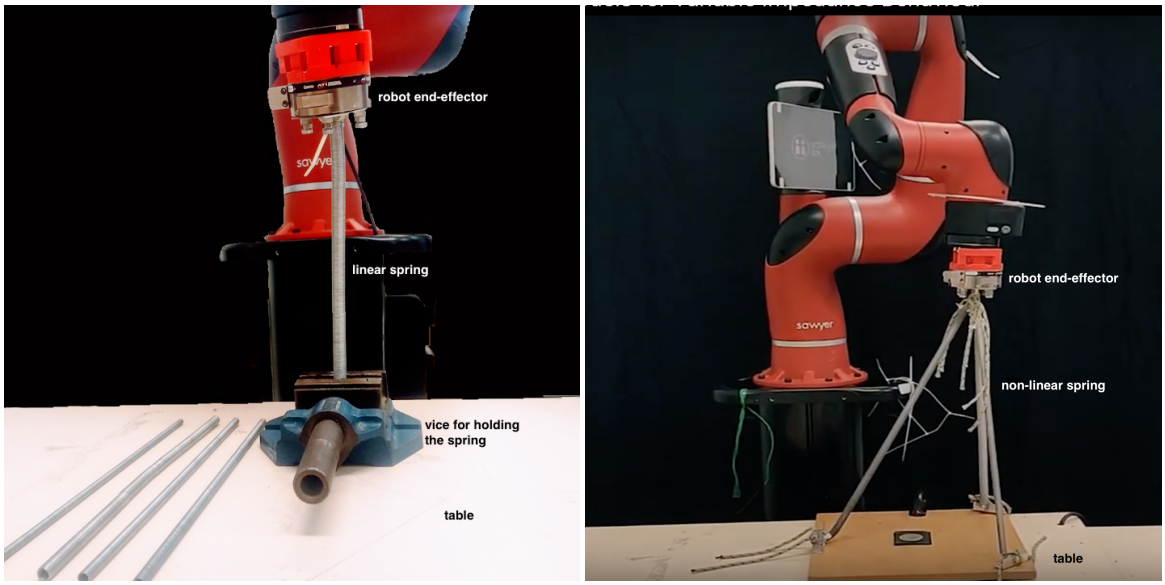


Figure 5.1: Spring pulling task setup. The springs are attached to the end-effector of the Sawyer robot and the other end is fastened to the table. Left: Linear spring (constant spring constant); Right: Non-linear spring (non-linear spring constant)

The primary goal of the task is to evaluate **H1** by moving the robot end-effector along a specified trajectory. The robot is attached with a spring (either linear or non-linear) at the end-effector with the other end fastened to a rigid environment.

The task setup is as shown in Figure 5.1. The non-linear spring used consists of a combination of three linear springs of different (B.1) spring constants connected at an angle. This spring arrangement produces a non-linear restoring force when it is pulled linearly. However, the robot controller do not know the parameters (e.g. stiffness, damping) of the spring attached to its tip. The other end of the spring is attached rigidly to the environment. When the robot tries to move towards the target position the spring starts extending, thereby pulling the manipulator in the direction opposite to spring extension. For safety purposes, the restoration forces produced by the springs on extension lie within the task space force limits (50N) of the manipulator.

The first task involved pulling the springs to a particular height and then moving along a desired trajectory. This experiment was conducted first with a linear spring (as proof-of-concept; Figure 5.1 left) and then with a non-linear spring (Figure 5.1 right), which is significantly more challenging due to its non-linear force response to extension. When the end-effector moves, it now experiences different changes in force in different directions. The baselines for comparison used constant low impedance and constant high impedance. The low

impedance parameters were chosen to be enough to move the end-effector along the desired trajectory in the absence of springs ($\mathbf{K}^p_{free}$), and the high impedance parameters ($\mathbf{K}^p_{max}$) were chosen to be sufficient for pulling the spring in the absence of the forward-model.

Observations

The existence of the spring is unknown to the robot controller. When the end-effector moves, the spring starts pulling in direction towards the other end of the spring. The end effector experiences force changes depending on the spring extension. Compared to the linear spring setup, the robot experiences different forces in different directions for the non-linear task setup. The constant low stiffness makes the robot highly compliant and hence results in the inability to overcome the restoration force of the spring leading to failure of achieving the target height. Repeating the task at higher stiffness [2] showed that the robot is capable of moving to the target height with the specified spring load.

After creation of the baseline, the task was repeated for following the trajectory using the control Equation 4.1, using the feed-forward term from the forward-model and impedance adaptation as described in Section 4.2.2. The task involved pulling the spring along the desired trajectory multiple times, without any previous training. Here, the framework starts to learn the forward-model of the task from first lap of the trajectory and then improves online in the subsequent rounds. The online learning helps the agent to learn a joint distribution of the robot's task-space force and velocity collected using the the end-effector sensors. The forward-model learned anticipates forces at each state thereby helping the robot to move closer to the target height. Yet, in some of the states the anticipatory control falls short of the target waypoint due to the inaccuracy in force prediction. This error is dealt with by adapting the stiffness with respect to the force prediction inaccuracy and further improves the tracking performance.

Results

	X (m)	Y (m)	Z (m)
no models	0.017 ± 0.009	0.015 ± 0.009	0.038 ± 0.010
high stiffness	0.012 ± 0.011	0.009 ± 0.007	0.023 ± 0.011
with learning	0.010 ± 0.010	0.006 ± 0.00	0.004 ± 0.00

Table 5.1: Experiment : Non-linear spring pulling. Trajectory tracking root mean square errors along the three axes along with its standard deviation.

[2] High stiffness parameters in the controller makes the manipulator stiff and robust to external forces

The results for the non-linear spring task are shown in Figures 5.2 and 5.3 (results of linear spring task are not included for brevity and due to result redundancy). Figure 5.3 shows the accuracy of the forward-model as it executes the task. The prediction improves significantly as the model gets additional experience compared to the first task trial. The experiment above

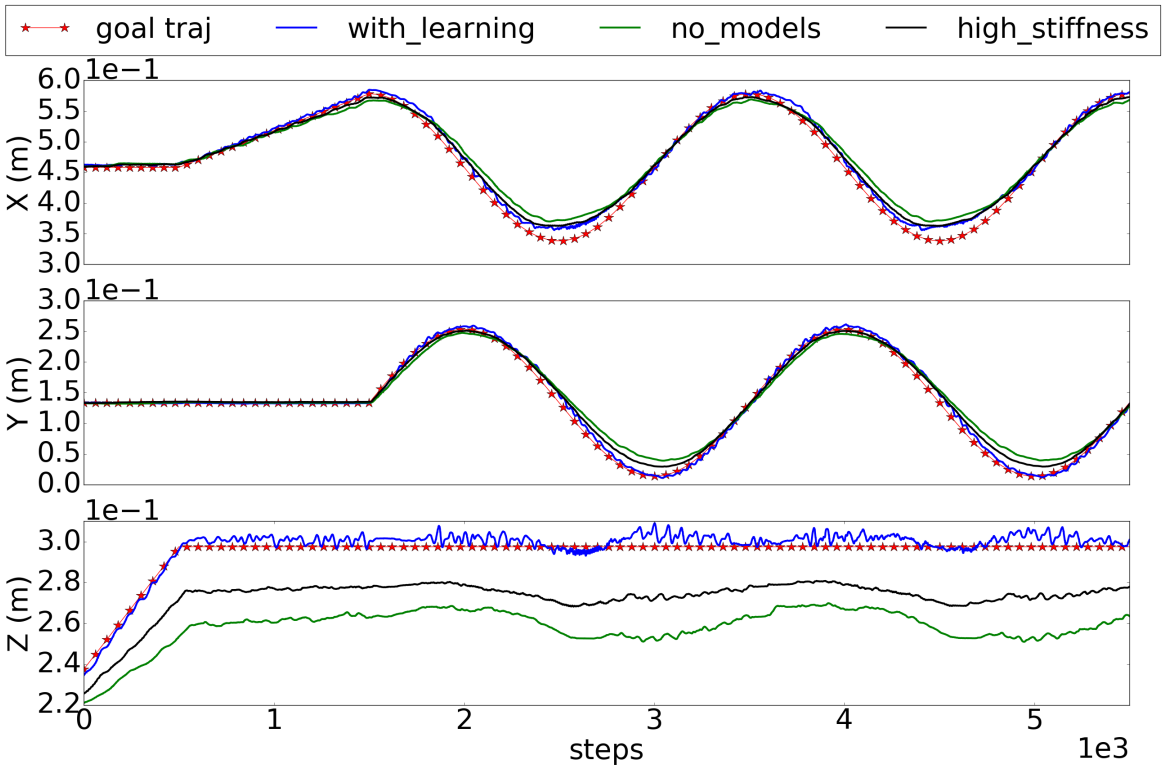


Figure 5.2: Experiment : Non-linear spring pulling. Three axis position tracking of the non-linear spring pulling experiment. The task requires the robot to maintain a target height and draw a circle. The position tracking errors are quantified in Table 5.1. The interesting plot is on the Z axis. We can see that our controller is able to maintain the target height far better than the high stiffness controller. This is because of the feed-forward term which act like a bias correcting term like the integral action of a PID controller.

shows existence of a forward-model under low stiffness helps the robot to be compliant with less trade-off to task accuracy. Subsequently adapting the stiffness of the manipulator at each state of the robot further improves the model and task accuracy.

Figure 5.2 shows that the accuracy of position tracking is better when the forward-model is used. The performance further improves significantly by varying the impedance parameters following the model accuracy. It can be concluded that using just the feed-forward term is not enough to perform the task accurately unless the learned model is perfect. Hence, the proposed framework improves the model online and adapts the impedance parameters depending on the model accuracy. We conducted multiple trials with the forward-model

being learned in the first trial and then improved in subsequent trials. The results are shown in Figures 5.2 and 5.3. Figure 5.3 shows that the prediction accuracy of the forward-model improves significantly over the trials.Figure 5.2 shows that the accuracy of position tracking and Table 5.1 quantifies the performance. Accuracy is better with the forward-model than

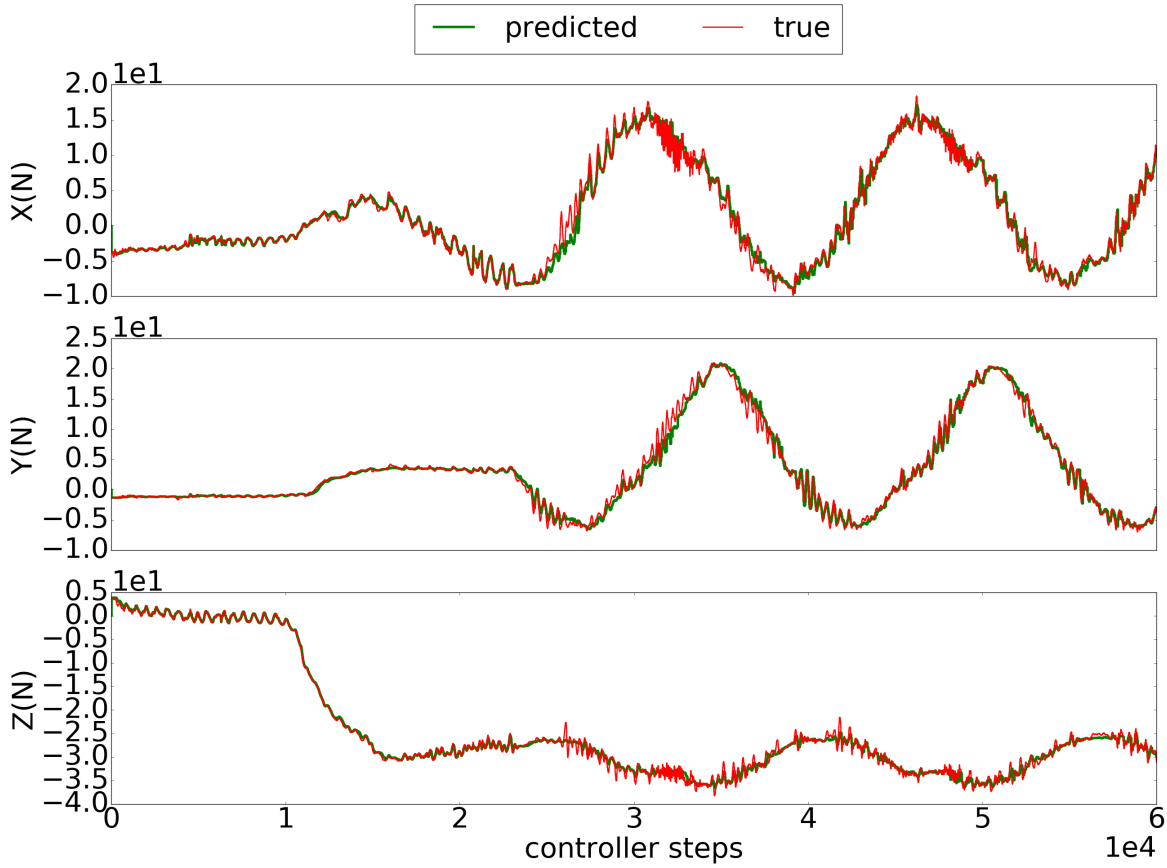


Figure 5.3: Experiment : Non-linear spring pulling. Force prediction of the three axis in the non-linear spring pulling experiment. The predicted values leads the true values and hence can act as feed-forward term to perform anticipatory control

when it is not used. Performance improves further when the impedance parameters are updated online; just using the feed-forward term is not enough to perform tasks that involve unexpected forces acting on the system (unless the model is perfect). The proposed framework improves the model online and adapts impedance parameters to the accuracy of the model.

5.3 Polishing different boards

To further test **H1**, and evaluate **H2** and **H3**, we explored the board polishing task; the robot had to wipe a surface of unknown friction coefficient. The task setup is shown in Figure

5.4. The robot wipes the surface by moving the rigidly attached sponge attached to its end-effector on the board surface. Two types of boards are used in the experiment. The first board is a white board used in lectures and the second is a wooden board. The white board relatively has lesser friction compared to the wooden board. The boards are attached to the table using clamps to arrest any possible motion when they are rubbed hard by the robot. In order to wipe the board the robot should apply a downward force and move on the surface. The motion on the surface of the board is pre-defined using a trajectory defined along the surface (the X-Y plane). For our experiments we chose a constant downward force of 10N. A hybrid force-motion controller is used since we can clearly segregate force and motion direction. The vertical downward force is controlled by the force controller and hence it provides compliance along the surface normal of the board (the z-axis).

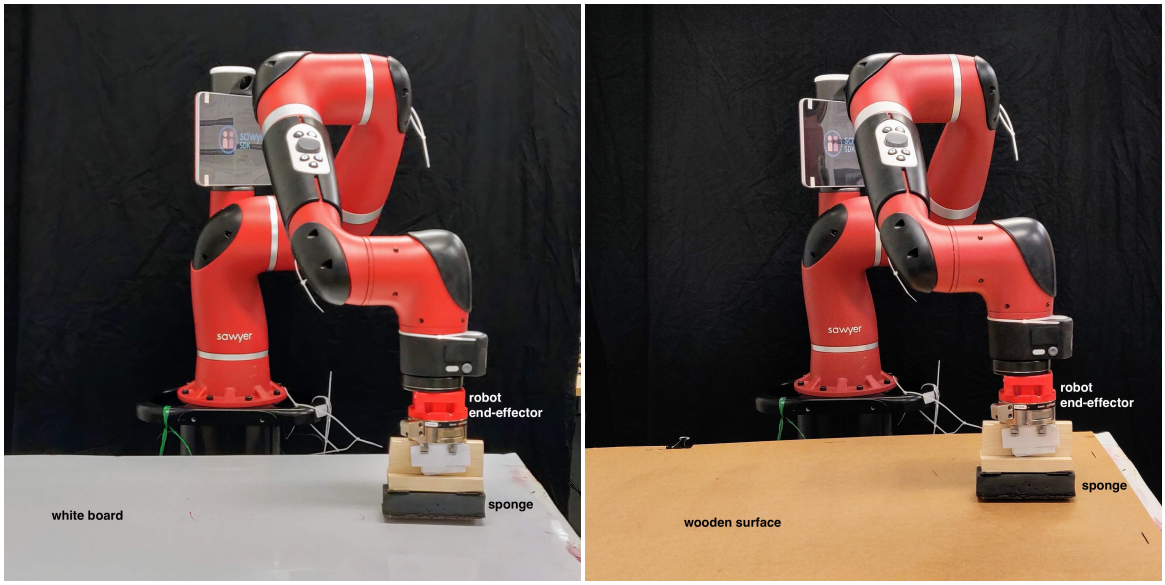


Figure 5.4: Board-Polishing Task Setup. Left: Surface 1 (low friction); Right: Surface 2 (higher friction)

While moving, the robot experiences frictional forces hampering its smooth motion, which it has to learn to predict. The initial model is learned by making the robot wipe the surface following a trajectory (an epicycle) that is considerably different from the one it has to follow (a sine wave) during task execution. This experiment involves control of motion and force in orthogonal directions which are specified as a hyper-parameter to the hybrid-force-motion controller.

The ability of the hybrid force-motion controller to provide compliance along the surface normal is examined with the board polishing task. Specifically, we ran trials in which the board was moved up and/or tilted during task execution. These trials qualitatively showed

that the robot was able to provide compliance along the surface normal. However, we do not include any quantitative results of these trials.

Observations

Initially, the robot polishes a table surface using the specified sinusoidal trajectory with high and low stiffness while applying $10N$ downward force to create baseline cases. The high and low stiffness values show the best and worst results which form the baseline trajectory tracking accuracy boundaries. To evaluate **H1**, the framework had to learn the forward-model of the task. A set of data (end-effector force, position and velocity) collected by repeating the task in high stiffness is used to learn a preliminary model online. Subsequently, the end-effector stiffness of the arm is reduced and the forward-model starts to predict next state force based on current velocity and force. The predicted force is used as a feed-forward term to compensate the frictional forces experienced during the task. The tracking accuracy improves substantially compared to performing the task at a lower stiffness. As the task is repeated, the new data collected is used to further improve the forward-model. The Gaussian Mixture parameters are updated when the likelihood of the new measured forces cannot be explained by existing forward-model distribution.

To evaluate **H2**, we conducted trials of board polishing task with the efferent copy in the feature vector of the forward-model. The addition of efferent copy (task space force) adds three additional dimensions to forward-model's input feature vector. This increases the time complexity due to the higher dimensional covariance matrix inversion that needs to be perform for regression. Results indicate that there is no significant improvement in performance for the additional time complexity of computation. We believe this is because the forward-model is able to obtain enough information for force prediction from the current end-effector velocity and force, making the information encoded in the efferent copy redundant. This observed performance, and the fact that addition of new dimensions to the state-space makes the learning more computationally demanding, led us to *not* use the efferent copy in the framework for subsequent experiments.

To evaluate **H3**, we focused on the ability of the robot to generalise and adapt to similar task environments. Such ability of the framework helps to perform the task without much human intervention. Our results with the board polishing task indicate that the framework generalises across different trajectories since the model was learned using a trajectory different from that used during task execution. This is a key advantage of learning the forward models in task-space instead of joint-space. Further, adaptability to new forces was tested by performing the same (board polishing) task using a surface with notably different

friction coefficient. While performing the task, we observed that the framework was able to adapt the already learned model to the new surface quickly.

Results

Figure 5.5 shows that in the absence of the forward-model, the robot is unable to follow the desired trajectory since it does not know the interaction forces. The use of the feed-forward model improves tracking performance, with a further improvement when online impedance adaptation is used. The performance of the framework is comparable with that of a high stiffness controller while requiring much lower impedance parameters (Figure 5.8). The average RMS error in trajectory tracking are summarised in Table 5.2. These results support the validity of **H1**.

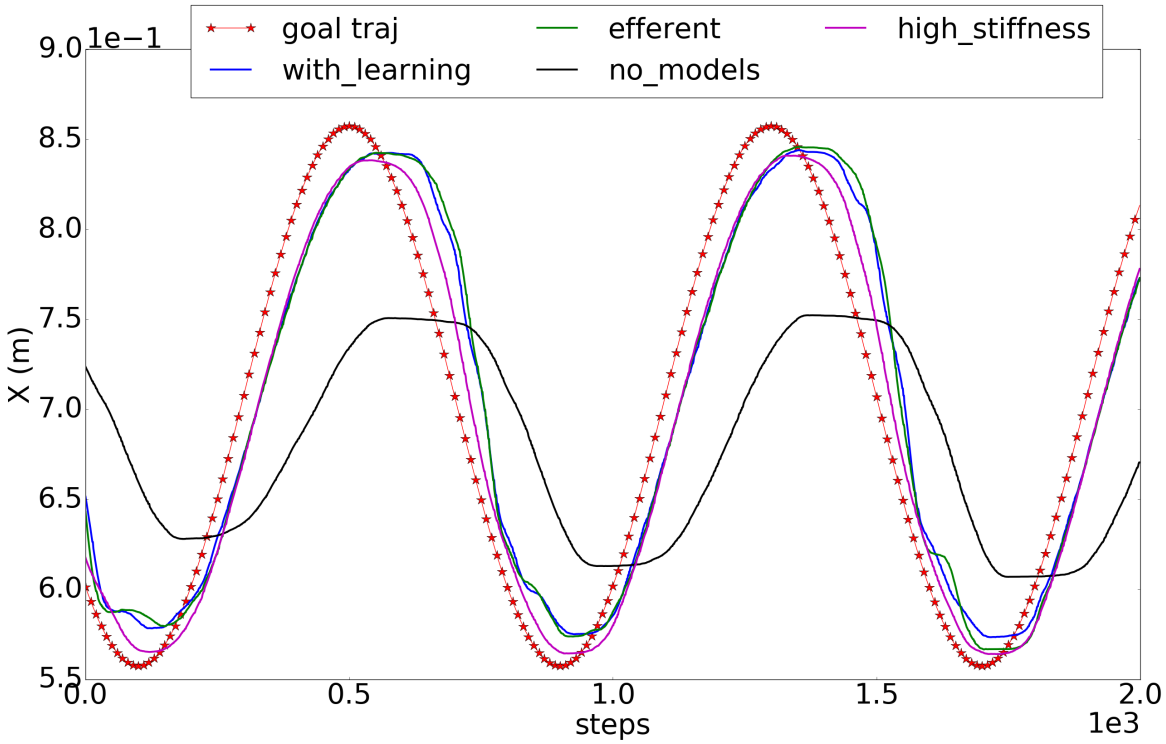


Figure 5.5: Experiment : Board polishing. Surface 1 position tracking. Red: Target; Pink: Using constant stiffness $\mathbf{K}^p_{free}$; Black: Using constant stiffness $\mathbf{K}^p_{max}$; Blue: Adapting impedance without efferent copy; Green: With efferent copy in feature vector

Figure 5.5 also further shows the performance of the framework when the efferent copy of control is used. The green curve, produced when the framework uses efferent copy in its feature component, is slightly better performance than the blue curve produced when no efferent copy is added. However, this improved performance comes at higher computational

cost compared to the forward-model which do not use the efferent copy. The reason could be the particular nature of this task where the feed-forward can be predicted accurately with robot state vectors alone.

	X (m)	Y (m)
With no models	0.091 ± 0.042	0.054 ± 0.010
Using high stiffness	0.027 ± 0.011	0.007 ± 0.014
With efferent	0.036 ± 0.024	0.008 ± 0.011
With learning	0.038 ± 0.023	0.008 ± 0.014

Table 5.2: Experiment : Board polishing. Trajectory tracking errors along different axes in case of the surface type 1 of board Polishing experiment. A visual representation of the errors can be seen in Figure 5.5.

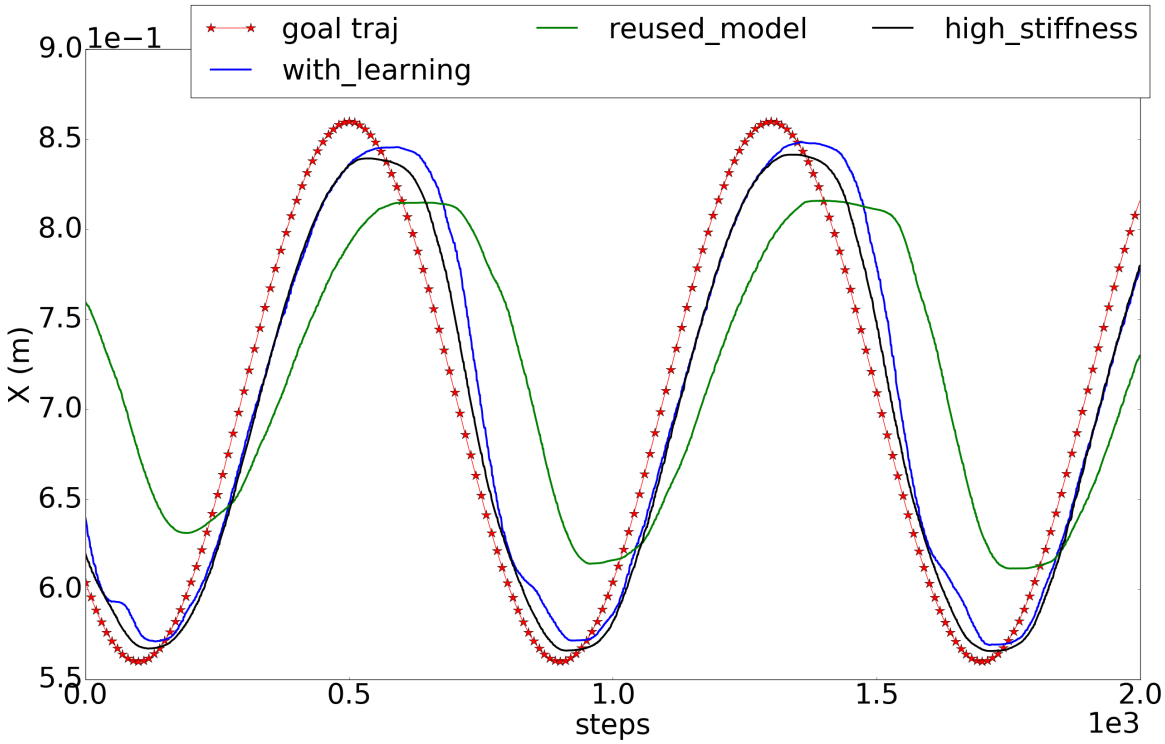


Figure 5.6: Experiment : Board polishing. Surface 2 position tracking. Red: Target; Pink: Using constant stiffness $\mathbf{K}^{p}_{max}$; Green: Using Surface 1 model without adaptation; Blue: Using online adaptation of previous model

Figures 5.6, 5.9, and 5.10 show that when the learned model was used for this surface without online improvement, the robot failed to follow the trajectory accurately. However, if the learned model is revised during task execution, it quickly achieves performance similar

to that with the first surface. The RMS errors in trajectory tracking are summarised in Table 5.3. This capability of the framework to generalise to different surfaces and trajectories is the critical advantage of using a *task-space, time-independent* variable impedance control framework.

	X (m)	Y (m)
high stiffness	0.027 ± 0.014	0.007 ± 0.012
reused model	0.084 ± 0.039	0.023 ± 0.022
with learning	0.035 ± 0.022	0.006 ± 0.001

Table 5.3: Experiment : Board polishing. Trajectory tracking errors along different axes in case of the surface type 2 of board Polishing experiment. A visual representation of the errors can be seen in Figure 5.6. The performance of the low stiffness controller with learned forward-model has an tracking accuracy similar to that of the high stiffness controller. It also shows that with an accurate forward-model the robot is able to achieve reasonable task performance at lower stiffness.

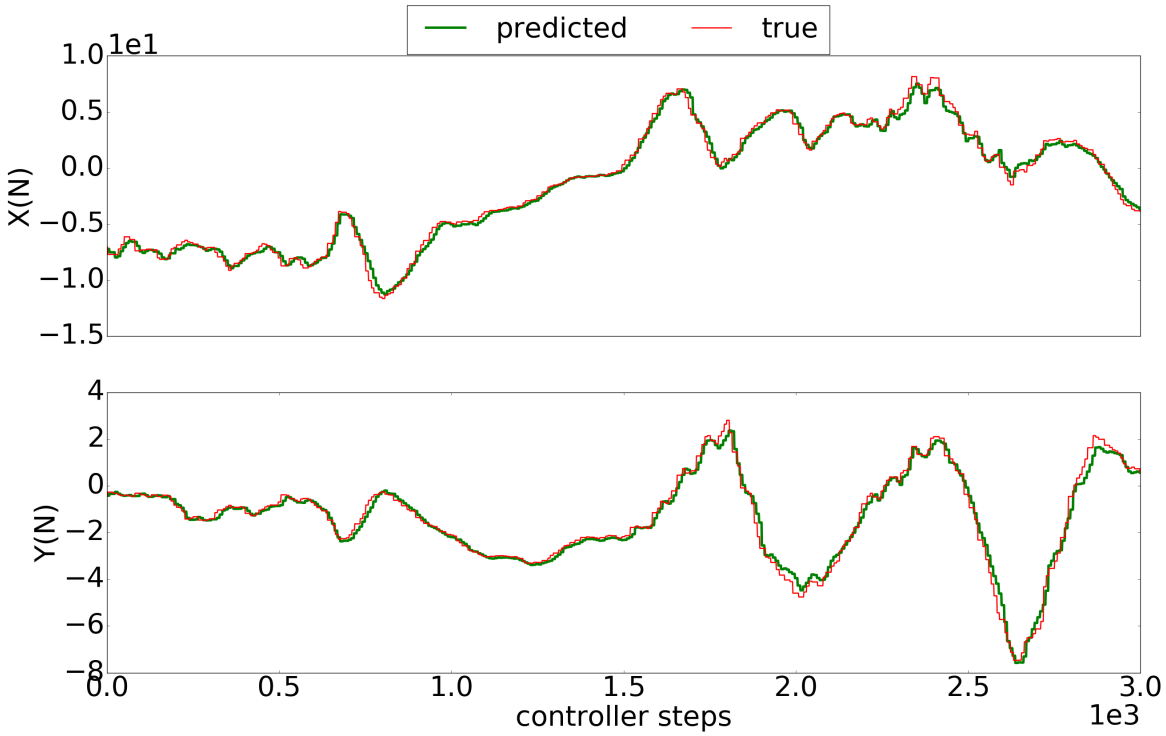


Figure 5.7: Experiment : Board polishing. Forces predicted by the forward-model for surface 1. It can be seen that the force prediction leads actual force measurement. The X axis consists of 3000 steps since the controller runs at higher frequency to the waypoint specification. The trajectory specified on the board surface is a sinusoidal wave. The error in force prediction is quite less and hence the stiffness adaptation in Figure 5.8 is more or less stationary.

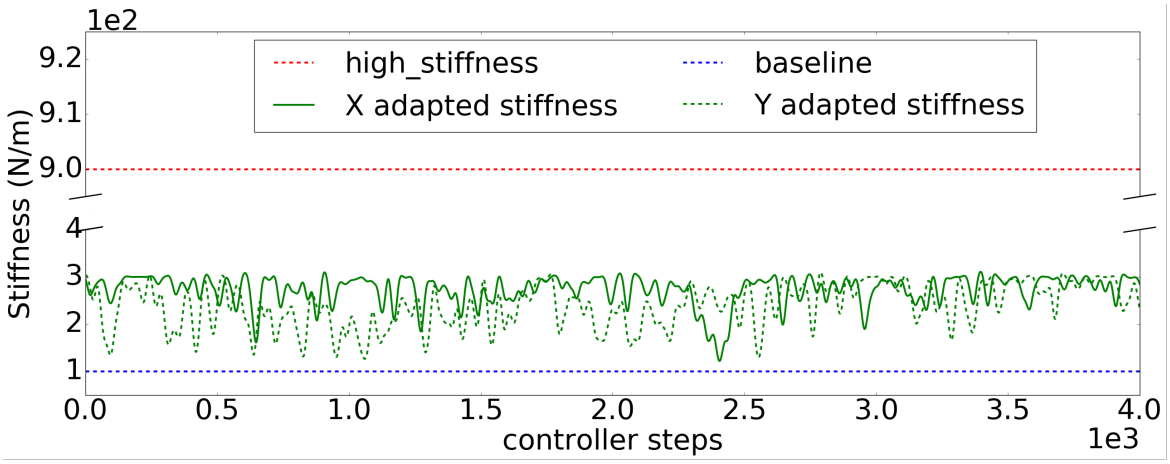


Figure 5.8: Experiment : Board polishing. Stiffness based on the forward-model uncertainty for surface 1. The pre-training of the model using the epicyclic trajectory described helps to learn a good forward-model. The surface is more or less uniform and the feature vector do not have the position component, the learned model is sufficient for other trajectories. The average value of stiffness along X axis is 275.54N/m and along Y axis is 225.82N/m.

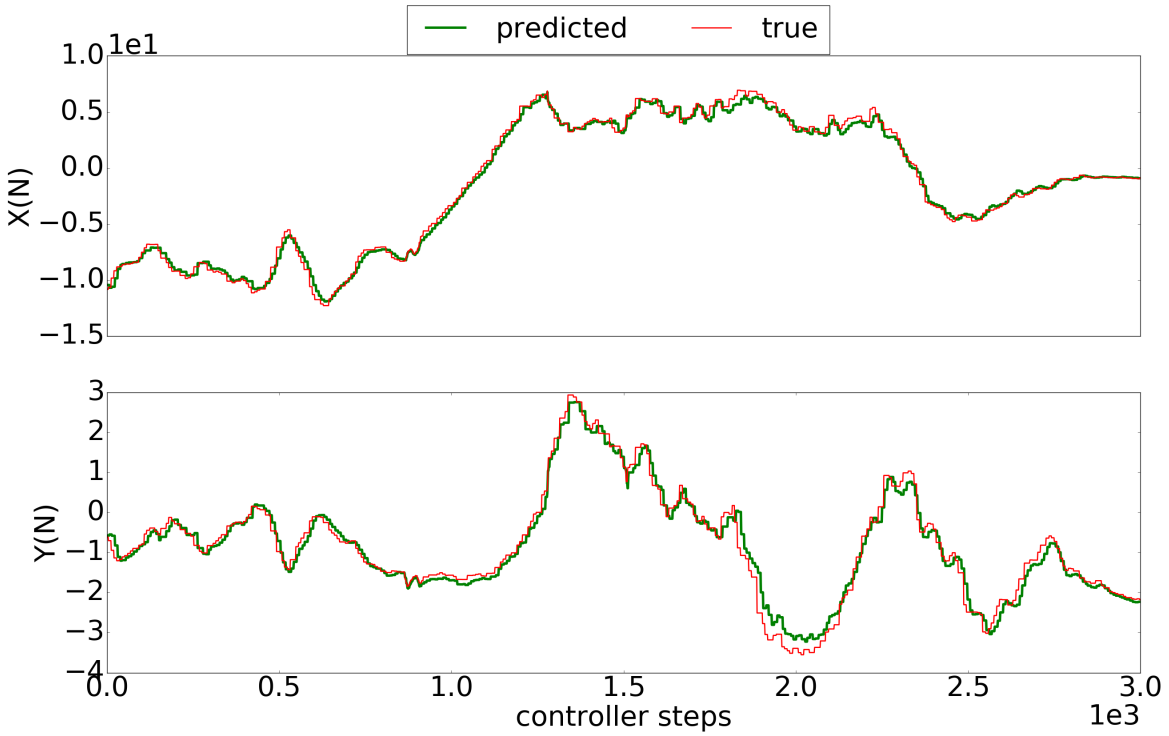


Figure 5.9: Experiment : Board polishing. Forces predicted by the forward-model for surface 2. This surface is rougher than the first surface. Here also, we can see that the prediction is leading the true values and thus helps in anticipatory control.

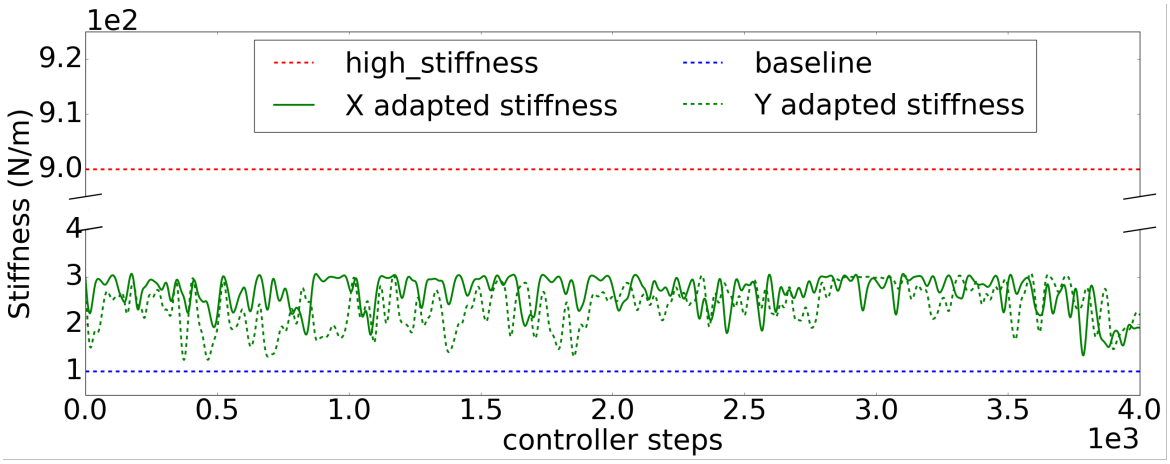


Figure 5.10: Experiment : Board polishing. Stiffness based on the forward-model uncertainty for surface 2. The stiffness adpatation of the new surface is comparable to the first surface. T

5.4 Porridge stirring

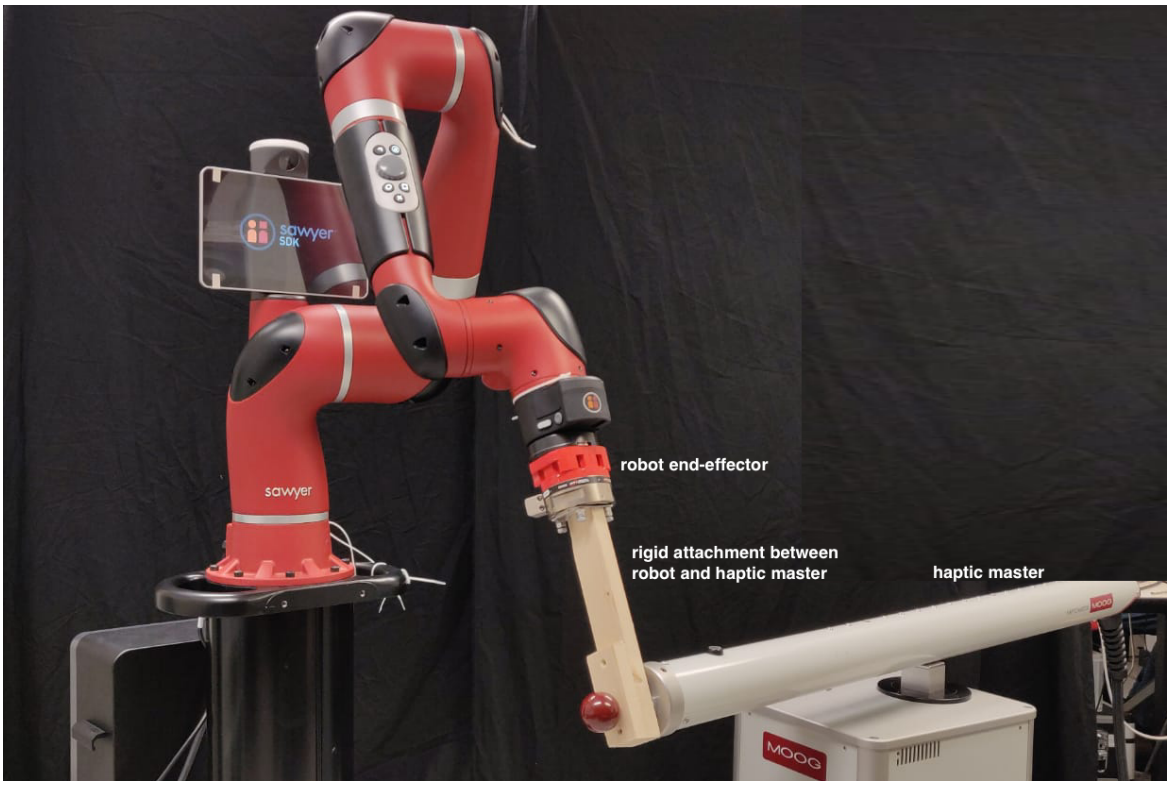


Figure 5.11: Porridge experiment setup. The porridge environment is emulated on the Moog Haptic Master. Sawyer's end-effector is initially aligned with the tip of the haptic device and connected using a wooden plank. The maximum drag force generated by the haptic device is withing the limits (50N) of Sawyer.

We consider an experiment to test the adaptability of the framework to environments that change during task performance with the porridge stirring task. The viscosity of porridge changes as it is stirred. For the experiment on a real robot, the changes in viscosity are emulated on a *MOOG HapticMaster* (Haptics [60]). The primary reasons for using this device is due to the difficulty in setting up an actual porridge stirring environment (e.g. safety issues within the lab) and since it gives options for accurate measurement of the environment properties (e.g. force felt, damping factor). These accurate measurements will help to validate the framework precisely.

For our experiment, the viscosity (damping factor) of the environment is increased continuously (in X-Z plane) until it reaches a maximum predefined value. The end-effector of the Sawyer is attached to the end-effector of the HapticMaster – see Figure 5.11. The Sawyer has to move its end-effector along a predefined motion trajectory (in the Y-Z plane) while learning to adapt to the increasing viscous resistance from the environment. As the robot turns the crank, the damping of the haptic master is programmed to increase exponentially with the velocity of stirring and time.

Observations

The robot is connected to the HapticMaster and the behaviour of the environment is unknown to the controller. The robot is required to follow a circular trajectory in *Y-Z* plane. The changing damping behaviour of the haptic device results in non-linear reaction forces on robots end-effector. When the damping varies as a function of time alone, the environment becomes intensely sluggish irrespective of the robot's motion. Moving in such an environment is quite tricky as the environment dampens even before the system is able to collect data. Hence, we varied the damping as a function of robot's time and stirring velocity. Here the damping of the environment increases whenever the magnitude of the velocity crosses a threshold. Similar to the previous experiments the baselines are established by performing the task at high and low stiffness. The tracking accuracy of the robot is depicted in the Table 5.4 and in Figure 5.13. The responses of the high stiffness trajectory tracking clearly shows the need of some additional compensation such as a feed-forward term in the control commands.

The compensation to the environment reaction forces is given by the feed-forward term. The system gets better with learning the forward-model of the task. The system is able to adapt to the incrementally changing environment and hence provides good evidence for **H3**.

Results

	X (m)	Z (m)
no models	0.031 ± 0.015	0.035 ± 0.018
high stiffness	0.012 ± 0.009	0.014 ± 0.012
forward-model	0.037 ± 0.017	0.026 ± 0.010
forward-model with stiffness adaption	0.014 ± 0.008	0.009 ± 0.007

Table 5.4: Experiment :Porridge stirring. Root mean square trajectory tracking errors along with the standard deviation of the porridge stirring experiment.

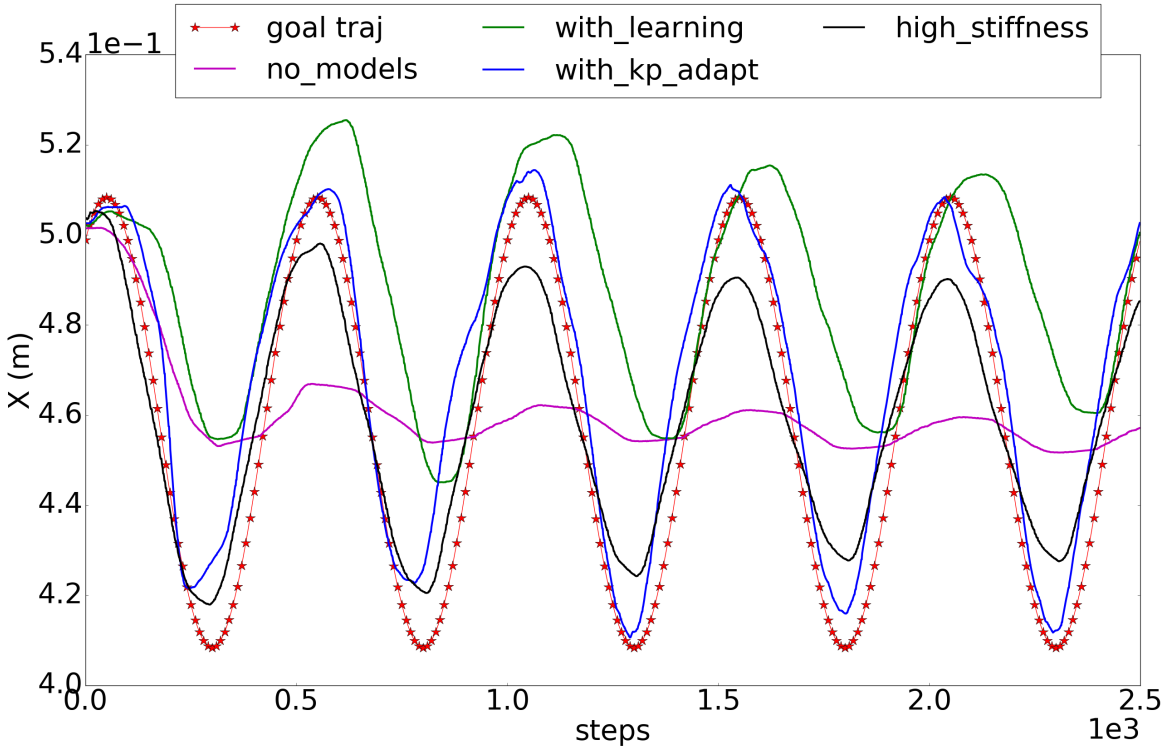


Figure 5.12: Experiment: Porridge stirring. Position tracking along the X axis.

The robot almost did not move in the low stiffness case due to heavy damping of the environment. The results of the previous experiments show that the high stiffness response is good in terms of the tracking accuracy. However, in this experiment, the high stiffness wasn't able to produce good results. Figure 5.13 indicates that using constant impedance parameters is not enough for tracking the desired trajectory (shown in red) accurately, even with the maximum allowed stiffness $\mathbf{K}^{p}_{max}$ (black). The reason is insufficient force applied by the robot and lack of the feed-forward term to compensate the forces.

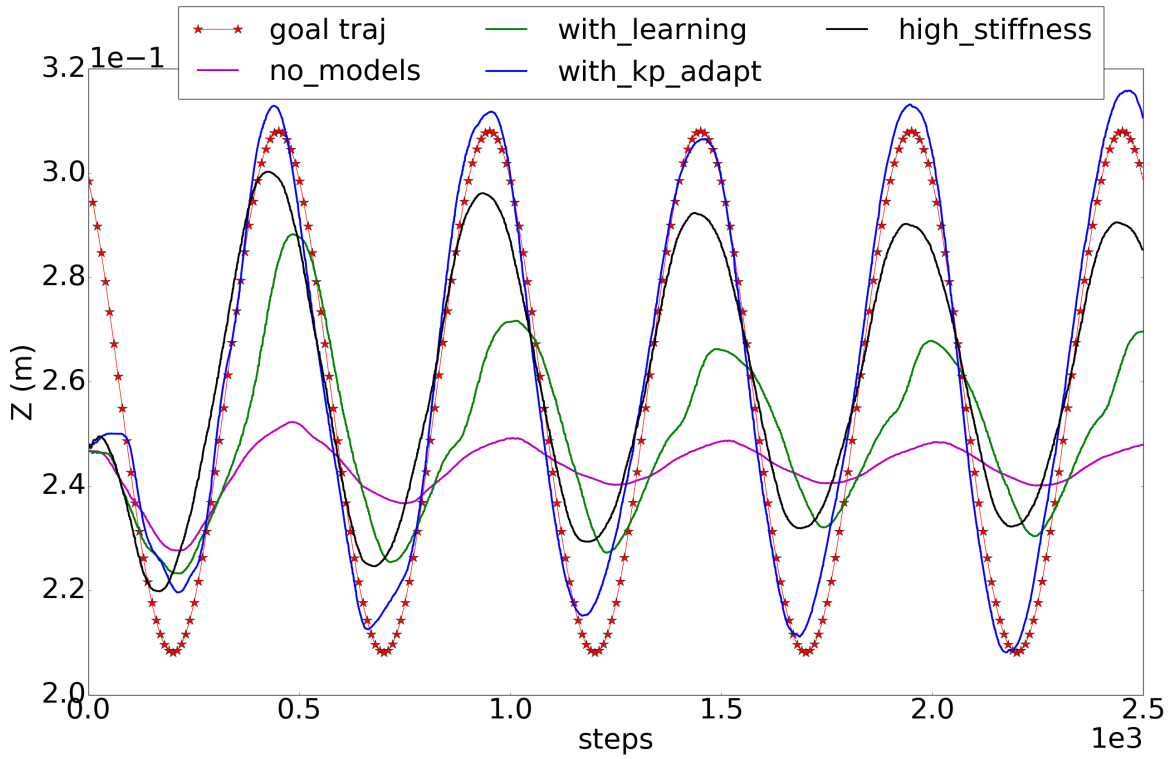


Figure 5.13: Experiment: Porridge stirring. Position tracking along Z axis.

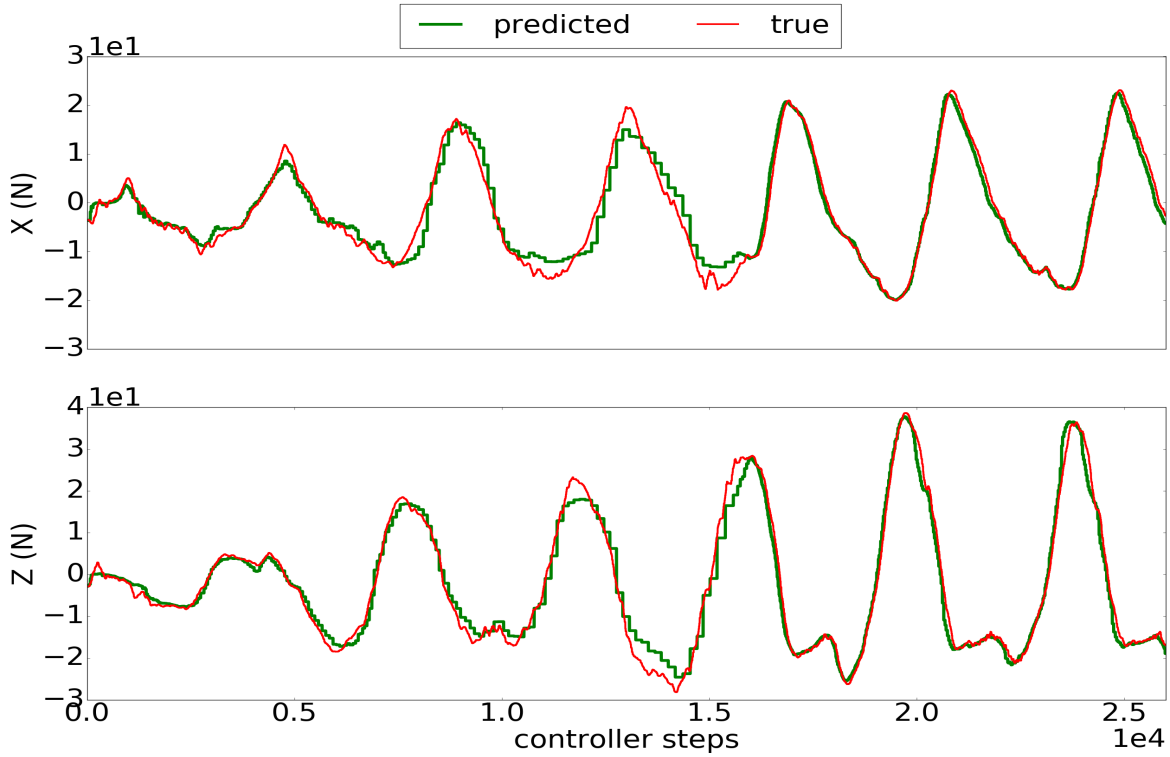


Figure 5.14: Experiment: Porridge stirring. Force prediction the porridge stirring experiment.

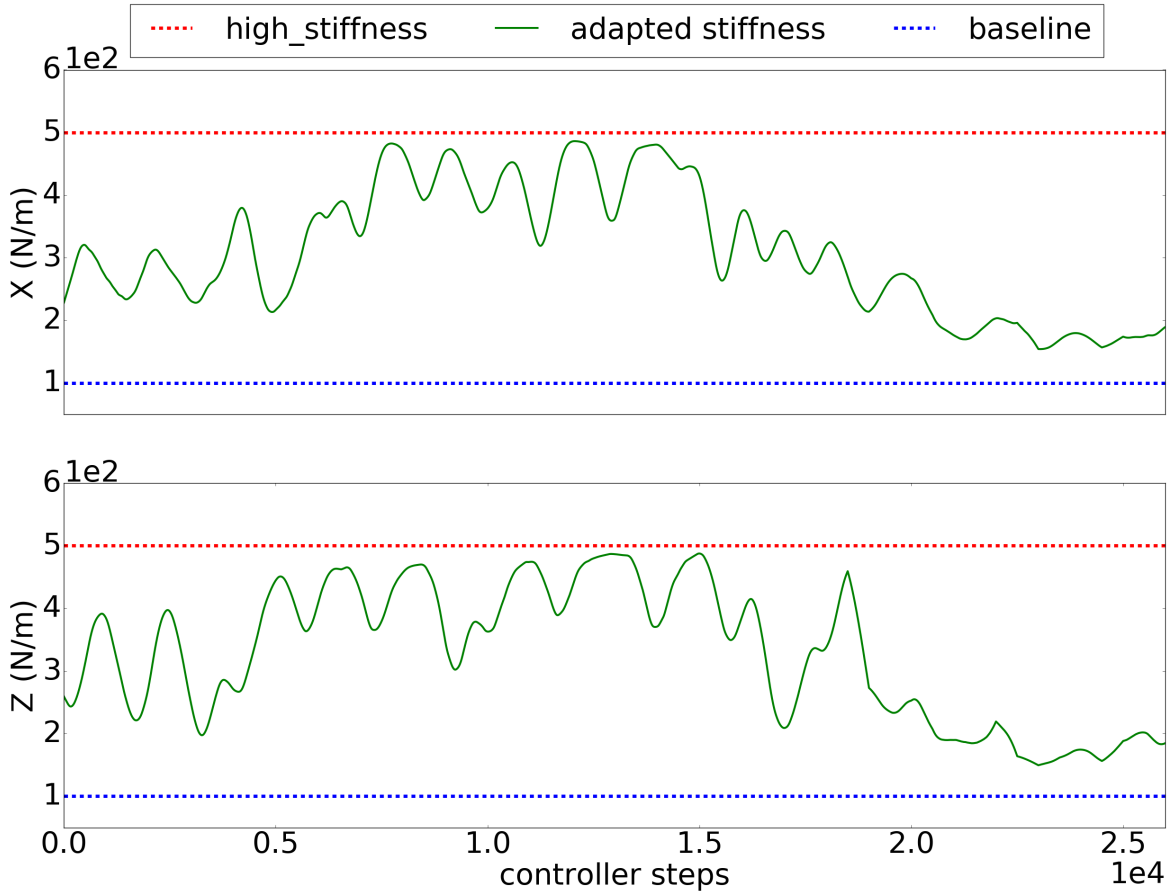


Figure 5.15: Experiment: Porridge stirring. Stiffness adaptation of the porridge stirring experiment.

Similarly, using the feed-forward term predicted by the forward-model alone without impedance adaptation is insufficient for such dynamically changing environments (shown in green), owing to the dynamics of the environment that are unknown to the robot. However, when coupled with the online impedance adaptation, the robot is able to follow the trajectory more accurately (shown in pink). The trajectory tracking errors during task execution (see Table 5.4) further establish the effectiveness of the framework. Next, Figure 5.14 indicates that the force predictions (by the forward-model) are much more accurate towards the end of the task when the environment finally becomes static as the change in damping saturates. The model has (by then) learned to accurately predict the effects of that damping factor on force. As a result, the impedance parameters take smaller values, as shown in Figure 5.15. Note that the stiffness values are still lower than the high stiffness value ($\mathbf{K}^{p}_{max}$).

5.5 Discussion

Variable impedance is vital for reliable and safe contact manipulation task. Learning impedance parameters directly is difficult and would require large training data. Using motivations from human motor control, we presented a framework that can learn forward-model of a task online, which can predict interaction forces in a state-dependent fashion. The accuracy in prediction is used to vary the impedance and improve the model online during task execution. The framework tests three hypotheses using three sets of tasks: spring-pulling, board-polishing and porridge stirring. The linear spring pulling assessed viability of the framework. Studies like Fan et al. [46], Liu [99] also use predictive models created using knowledge of mechanics to make predictions about robot and object motions. However, unlike our method, they make unrealistic assumptions such as point contacts, polygonal approximations to the friction cone and lack of slip during motion. Success of the framework in the linear spring pulling motivated us to apply the same framework to more challenging non-linear spring pulling task. However it is observed that, just employing forward-model wasn't sufficient to get satisfactory results. This provided insights to use a state-dependent stiffness controller whenever the forward-model had to face inexperienced forces. Though the technique proposed in Kronander and Billard [81] is able to learn a state dependent stiffness controller, it relies heavily on learning from demonstration through a specially designed expensive hardware.

Another challenge to be addressed was the choice of feature vectors required to represent the task state. The board polishing task tested different types of parameterisation as explained in Sections 4.2.2 and 5.3. The use of robot end-effector velocity instead of position as a feature in the state vector helped to generalise across different trajectories and surfaces. Using position vector as a component of the feature vector will cause the learned model to be spatially dependent. This will mean that even if the surface is similar, the robot will have to experience every part of it to learn a state dependent stiffness controller. Approaches such as Buchli et al. [19] and Deniša et al. [38] lacks this generalisability due to their explicit time dependence for representing force and impedance parameters. The ability to generalise also shows the effectiveness of learning forward-models in task space rather than in joint space such as in Kronander [82], Kalakrishnan et al. [71, 69]. Other variable impedance works such as Schneider and Cannon [141], Wimböck et al. [167], Li et al. [94] requires accurate mathematical representations of the task, robot and the objects involved. The proposed framework do not require such analytic models for control.

The porridge stirring task involved a dynamically changing environment and was the most difficult of all three experiments. Such an environment whose viscosity changes depending on robot motion is unique compared to other popular tasks in the literature. This task

provides a good example of a controlled environment to emulate and implement dynamically changing environments. The learning process becomes more challenging since the force experienced is a function of the end-effector velocity rather than its position. The successful stiffness adaptation with improved task performance provides strong evidence about the capability of the framework. Some approaches like Kronander [82], Huang et al. [65] try to learn an action-effect correlation, usually from demonstrations provided by an expert or from experience derived from several trials (Levine and Koltun [91], Kupcsik et al. [85]). However, it is difficult to teach continuous interaction tasks to robots without additional hardware. The experiments conducted indicate the ability of the framework to learn and use time-independent variable impedance controller. The controller is defined in the task-space and works across diverse tasks without any modifications.

The controller defined also assumes the existence of a constraint between the proportional and derivative gains given by equation 4.3. Though this condition was required for the safe interaction of the robot in the tasks, it resulted in producing a temporal lag in the controller response. This lag can be seen in the trajectory tracking plots of all the three tasks. Apart from the lags in the response, the key insight derived from the experiments is that forward-models are not alone sufficient to perform the task better. The forward-model should be used in conjunction with the feedback to have the best performance. But in this way one can argue that the proposed approach is exactly similar to the adaptive control methods. However, our method performs an online learning of the forward-model which helps in quick adaptation of parameters for similar tasks.

Technical challenges

Initially the framework implemented was tried using a single machine running with Ubuntu 16.04 and ROS [3] Kinetic. Running online learning, control loop, trajectory waypoint thread, and sampling from force torque sensor together on the same machine made the process slow and inefficient. For example, the online learning involved inversion of the Gaussian Mixture for prediction which has a time complexity of O(NKD3)[4]. We know that, implementing impedance control requires direct torque control of the robot. The Sawyer robot consists of a real time loop to sample the torque commands from the commanding PC. The inbuilt controller of the Sawyer expects the control command frequency to be a minimum of 500Hz. However, the overloaded PC was unable to maintain the required command frequency and

[3] Robot Operating System

[4] where N is the number of data points, K is the number of Gaussian components and D is the problem dimension.

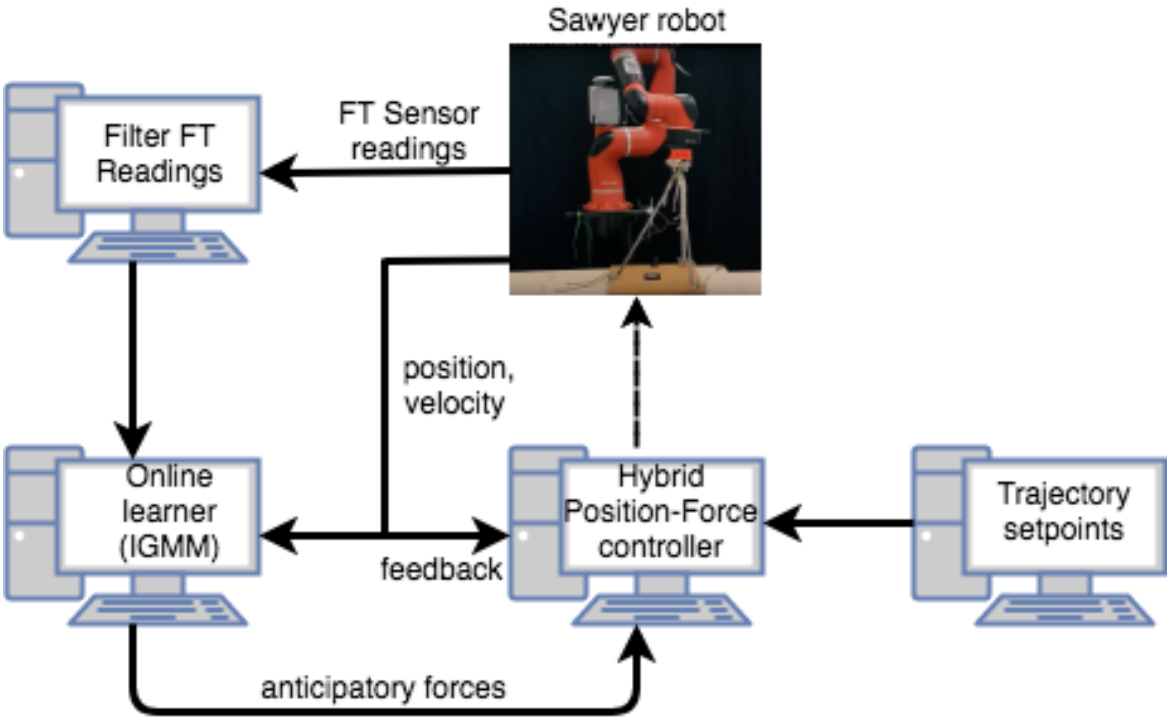


Figure 5.16: The setup of the network architecture. All computers run ROS Kinetic on Ubuntu 16.04

hence resulted in sluggish robot behaviour. These problems motivated us to implement a network of computers as shown in Figure 5.16 to perform the task.

ROS uses messages to communicate between computers, robots and other devices. These packets by default uses TCP [5]. When TCP packets are sent to the robot, if some packets are lost, the host machine retries till the correct packet arrives. This results in incorrect behaviour of the robot if it is already in motion. Hence it is better to use UDP [6] packets when controlling the robot instead of TCP packets.

One of the severe challenges we faced during the development of the framework and testing of the experiments is a bug in the SDK of the Sawyer robot. Our robot runs Intera 5.2.1, and in this software, there is a mismatch between the end-effector frames. The data given by the SDK to our controller contains end-effector position in the base frame (global frame, right-hand coordinate system) of the robot, end-effector velocity in the end-effector frame (local frame, right-hand coordinate system) and the end-effector force in a left-handed end-effector frame. This mismatch in frames always triggered an incorrect response of the robot. It took several months for us to figure out this undocumented bug in the SDK.

[5] Transmission Control Protocol
[6] User Datagram Protocol

Chapter 6

Conclusion and future works

We started this research in the domain of dexterous manipulation motivated by the phenomenal dexterity of human hands. Strong evidence from human-motor control studies explained in Chapter 2 provide insights into the importance of having task-specific forward-models in our motor control. This thesis primarily investigates the use of similar forward-models in robot manipulation. Promising results of similar studies in robot manipulation research further support the approach. In this chapter, we summarise our motivations, contributions and their inherent limitations. Some of the points mentioned in the later future research section are to address these issues while others are new directions of potential research.

6.1 Forward-models for push planning

Initially, planning a sequence of actions using forward-model was identified to be the fundamental problem to perform robot manipulation. As a first step, we decided to pursue the problem of push manipulation due to two reasons: (1) motivation from the fact that pushing is a primitive and widely used technique for prehensile manipulation by humans (2) lack of sufficient literature showing methods to perform robust push manipulation with robots. Planning robust actions require the knowledge of the object's behaviour under an action. Object's dynamics dictate this behaviour and can be obtained via forward-simulating the object's dynamic model. However, prediction of the object behaviour under an action has uncertainty. This uncertainty arises due to noise in the environment or having incomplete knowledge of the dynamics of the object. The scope of addressing the issues due to environmental noise is limited. The incompleteness in the dynamics model could be from (1) approximations and assumptions made during mathematical modelling or (2) from insufficient data in all parts of the object's state space. One of the reasons for the lack of robustness in robot push

manipulation is the inability of the existing push manipulation planners to consider this uncertainty.

Push planning without considering uncertainty can result in the object getting lost in areas of high uncertainty. This can be seen in our experiment result shown in Figure 3.8b. And hence in our approach, we decided to use forward-models which can learn from data (due to the reasons explained in Section 3.2) as well as provide a measure to its predictive uncertainty. Including knowledge of predictive uncertainty helps in planning actions which upon execution moves the objects through less uncertain areas of object space. We included predictive uncertainty in planning by combining learned forward-models with stochastic planners. We propose and test two versions of a planner to generate a sequence of uncertainty averse pushes to move the object through less uncertain areas towards the goal pose. The proposed two planners form the first contribution of this thesis. Chapter 3 details the complete details of them.

The first version of the planner explained in Section 3.2.2 optimises a combined cost function given by equation 3.11. The cost function consists of a linear combination of goal cost and an uncertainty penalisation. The planner generated push actions depending on the weights. The relative value between the goal cost and the uncertainty cost decides the push actions. Higher weight given to the goal cost motivates the planner to move faster towards the goal at the expense of getting lost in uncertain regions. A higher cost on the uncertainty penalisation can move the object in the least uncertain regions and thus take a long time to reach the goal. Finding the right balance between these weights is quite tricky and requires careful hand-tuning. The results of the planner using a forward-model (E-MDN) learned from a toy dataset after careful hand-tuning of weights is shown in Figures 3.9a. The figure shows that the planner can push the object towards its goal pose through regions of less uncertainty and thus provides evidence for our hypothesis **H1** (refer Section chap3sec:experiment). From the figure, we can observe that the planner does not move the object through the blue regions where the forward-model is fully certain. This is because moving through those regions will result in a longer path towards the goal pose. The right balance between the goal cost and the uncertainty cost helps the planner to decide what level of uncertainty will not hurt the planning. This consideration helps it to find the optimal push actions to move the object in the shortest path possible. Though this planner was a success, it took a considerable time to hand craft the right weights for the cost function. Even with the right cost function, the optimiser takes a long time to converge. This is because the optimiser has to come up with a solution that satisfies two opposing cost functions. The second version of the planner was an attempt to solve these problems using hypothesis **H3**.

In the second version of the planner, we attempted to separate the two components of the cost function. Since the goal is to move the object through regions of less uncertainty, we can first generate an object path through the less uncertain regions. Later this path can be used by the planner to push the object towards the goal. Initially, we start with a straight-line path between the start and goal pose of the object. Later this path is updated incrementally by using randomly generated trajectory samples in its neighbourhood. These trajectory samples are weighted using the uncertainty cost function. The samples passing through areas of high uncertainty get penalised more than the ones passing through areas of less uncertainty. These weighted trajectory samples help to compute the gradient of uncertainty. The gradient helps to shift the initial trajectory in the direction of least uncertainty. The optimiser stops modification of the trajectory once the gradient settles to a constant value. Subsequently, the first version of the planner uses this less uncertain trajectory to find optimal push actions. The planner takes each point in the trajectory as intermediate goal points and finds push actions to move the object towards that point. Once the object reaches within a threshold, the next trajectory point becomes the goal. In this problem set up, the optimiser has no notion of balancing goal cost and uncertainty cost. Hence, the amount of uncertainty the system can take becomes a factor to be determined. However, the splitting of the cost function into two helped the optimiser in spending less time to find a solution which balances these opposing terms. The result of the pushes generated by the second approach shows that they take almost half the time as the first and thus proves hypothesis **H3**.

Further to test the effectiveness of using learned models for robot manipulation, we used two forward-models trained from real data collected by the robot through the experiment detailed in Section 3.3. The performance of the learned model in comparison to that of a physics engine based model is studied. The two learned models used are Gaussian Process (GP) and E-MDNs. The results in Figures 3.10, 3.11 and 3.12 shows that the learned models are better than the physics engine based models. This change in performance is primarily due to these models' ability to capture environment interactions better than the physics model. Within the learned models, E-MDNs were better than the GPs and provided evidence for hypothesis **H2**.

In short, we use a forward-model (E-MDN) and its predictive uncertainty to perform risk averse push manipulation. We developed two algorithms to plan a sequence of risk averse push actions and is described in Chapter 3. These actions are capable of moving an object towards a target pose through known regions of a table's surface. The planner uses the receding horizon approach and hence allows to incorporate feedback to deal with model inaccuracies and unseen disturbances. The first algorithm optimises a cost function to simultaneously generate risk averse push sequence and object trajectory. The second

algorithm solves the same problem in two stages while consuming less time. First stage finds a risk averse trajectory and the second stage generates optimal push actions to move the object on this trajectory.

6.2 Forward-models for variable stiffness control

The success of the push planners provided us with further motivation to use forward-models to aid dexterous manipulation. The task of push planning involved discrete contacts and quasi-static object behaviour. However, most of the robot tasks require continuous interaction with the object and environment. However, we know that it is difficult analytically model the highly non-linear interaction dynamics between the robot and its environment. An alternate and effective way to model this non-linearity is by using data and methods from machine learning - measurements of the force-torque sensors, robot encoders can be used to collect data when the robot interacts with its environment. In our approach, we propose an incremental Gaussian mixture model to model the interaction data. The data from the sensors get fed continuously into the model, which automatically decides the number of kernels required to represent data. Section 4.2.2 covers more information about the proposed approach. Each of the datapoints consists of [previous end-effector velocity, previous end-effector force, current end-effector force]. We use conditioning to use the learned model as a forward-model.

In order to predict the next step force, the learned model is conditioned on the current sensor measurements of end-effector velocity and force. This predicted force helps the robot to anticipate the interaction force and perform the corrective actions in control. When a robot interacts with an environment, there are forces to which robot has to be compliant, and there are forces to which the robot should be stiff. For example, in the board rubbing exampled described in Section 5.3 the robot should be compliant to a force disturbance in the Z direction. At the same time, the robot should be stiff to the frictional force, which impedes its smooth motion on the surface. Thus, the manipulator should act as an impedance in the Z direction and admittance in the X-Y direction. Hence we need to have variable impedance behaviour in different directions. It could also be the case that instead of directions, it could be at each state. For example, consider the porridge stirring example provided in Section 5.4. In this example, it is better to be stiff while moving the haptic master against gravity and compliant while moving down. Hence, we require variable stiffness depending on the state of the task. These are two different cases of variable impedance control. A case in which directions of stiffness and admittance are known and a case in which it is state dependant. In order to combine these two cases, we define a hybrid force-position controller (Section

4.2.4) in task space. (Note: a task space controller automatically provides some level of compliance in the joint space of a redundant manipulator like Sawyer.) The predictions from the forward-model anticipate the interaction force. A perfect forward-model will not require any stiffness adaptation since it can anticipate force at every task step. However, since it is an online model, the force predictions of the system are less accurate when the model does not have sufficient experience. To mitigate this, we propose a variable impedance behaviour depending on the error in force prediction. Section 4.2.3 covers further details on the topic. In short, the major components of the control framework are a forward-model learning module and a hybrid force-position controller.

The complete framework described in Chapter 4 is motivated by biological motor control theories. Strong evidence from human motor control studies detailed in Section 4.2.1 support the idea of the proposed approach. The framework is capable of variable impedance behaviour in continuous interaction tasks and thus constitutes our second contribution. The approach learns the forward-model of a task through repetition. The stiffness modulation happens as a function of model inaccuracy leading to a state-dependent impedance controller. This controller enables the robot to apply correct end-effector stiffness at each state of the task. The thorough study and experiments detailed in Chapter 5 prove the capability of the framework and forms our third contribution. The experiments considered deals with continuous interaction tasks in static and dynamic environments.

The results of the experiments show that the use of forward-model highly improves the tracking performance. The tracking accuracy of all three experiments strongly supports hypothesis **H1**. Further, we noted that the stiffness controller learned is transferable to other similar tasks. In similar tasks, for example, polishing a different board than the trained one, the learned forward-model can quickly adapt to the new surface properties. The difference in using the pre-trained model and learning from the beginning has a notable difference in the graphs. This shows strong evidence of hypothesis **H3**. However, it also has to be noted that in our particular setting learning the dynamics of Surface 2 using dynamics model of Surface 1 results in relearning of the model. The model adapted to Surface 2 will have to relearn again if applied again on Surface 1. Addressing this limitation is beyond the scope of this thesis.

In short, the force predictions of the learned model are used to perform anticipatory interaction control, and its predictive errors are used to control robot stiffness. Experiments such as board polishing presented a static environment, while non-linear spring and porridge provided dynamic environments. Board polishing task showed the adaptability of the learned model to new surfaces. The framework can anticipate forces in dynamic environments and showed better performance than the baseline tasks. The performance of the same

framework (with no modifications) on different tasks in diverse environments illustrate its generalisability.

6.3 Limitations and future work

In this section, we discuss different limitations of our contributions and thus limits their scope when applied to other scenarios. However, addressing some of the drawbacks discussed in this section can extend these solutions to broader problem contexts. Additionally, we also describe some promising future research directions to make our solutions more robust. Initially, we discuss three limitations of the contribution made in Chapter 3. Subsequently, we discuss another three limitations of the contribution given by Chapter 4.

Forward-models for push planning

Chapter 3 proposes two versions of a planner which can avoid regions of high uncertainty while pushing an object towards the goal pose. These planning algorithms relied on a forward-model based on Ensemble-Mixture Density Networks for object pushing. The model used in for the experiments is limited to predictions (three dimensions) of object pose in 2D. Theoretically, the forward-model is capable of learning larger dimensions. However, time constraints did not permit us to test large DoF models. It will be interesting to explore the dimensional limitations of the model (if any) when trained on high DoF data. The major challenge in training a large DoF output will be the tuning of hyperparameters. The experience so far with the networks suggest following approach. Initially take a subset of the training data and try to overfit the model. Capability to overfit denote the network to be sufficiently complex to represent the data. Failure to overfit suggest modification of network parameters like more depth layers, modification of learning rate or tuning of other hyperparameters of the cost function.

Further, since our problem was constrained to 2D, and since it was a proof of concept work, we limited ourselves to an action value of 1 dimension. The optimal push actions found by the planner are in terms of this 1D value. This value gets converted to a push action using a predefined transformation. Though the planner supports the parameterisation of continuous actions, they are limited to a single type of actions. Instead of push actions, it would be interesting to investigate how to add a different parameterisation to include other types of actions such as toppling, hitting and using environment constraints (e.g. pushing against a wall) to augment pushing. The availability of different types of actions will make the planner better equipped to deal with uncertainty.

Another shortcoming of the current approach is the lack of ability to forward propagate the uncertainty in prediction. This shortcoming is mainly due to the inability of the forward-model to include uncertainty in the inputs while forward simulating to predict the next state. Hence, the current push sequence generated by the algorithm assumes a quasi-static behaviour of the object. This is the reason for the inability of the planner to generate continuous push actions. Thus the object moves from start to target in a quasi-static manner. It will be interesting to use a forward-model which can also incorporate uncertainty associated with the current state in the input. Adding the capability to propagate uncertainty will help to plan multiple push actions simultaneously.

Forward-models for variable stiffness control

In Chapter 4, we proposed a framework to deal with continuous interaction tasks. The framework can incrementally learn the non-linear dynamics when the robot interacts with the environment. The learned model is used by the framework to anticipate force, and the force errors are used to modulate the stiffness behaviour. The variable impedance proposed deals only with the shaping of the stiffness and damping terms of the controller. However, effective impedance control requires inertia shaping as well. In our case, we use the assumption of constant end-effector inertia tensor. This assumption is valid for forces which cannot affect the overall inertia of the robot. However, there could be task states which affect the overall inertia of the robot. Extending the framework with varying inertia tensor will make it more generalisable. However, varying inertia tensor requires great care and precision to avoid unwanted behaviour of the robot. The stability conditions, sensing and choice of additional feature vectors required to add modulation of inertia tensor is a challenging and exciting direction for future research.

Further, we can notice that in our two surface experiment (Section 5.3), the agent tends to forget the forward-model learned on surface 1 when it is moving on surface 2 and vice-versa.This forces the agent to relearn the forward-model when used back on the older surface again. This problem can be solved by using a *context* to identify each of the surfaces. In robotics, context is defined as a feature which helps to understand or classify a situation sufficiently. For example, polishing two surfaces with distinct friction coefficients requires the same kinematic movement, but different force adaptation. The context can be a visual or contact feature of the surface, which helps in its correct classification. This allows 'switching' of the forward-models as and when required instead of relearning. At present, the online learning model of the framework cannot deal with switch models based on contexts. This is because the current feature vector does not have a component to represent task context. Context identification will also demand additional sensors (e.g. visual) or

inference capabilities from existing sensors. Interesting research will be to learn or detect contextual features to augment the existing framework. This addition will potentially help in auto-switching of the forward-models depending on the context.

Earlier, we mentioned that the implementation of task space controllers helps in introducing some amount of joint space compliance for redundant manipulators. The current framework deployed do not optimise any forces in the task space. This is because direct optimisation of the end-effector forces in task space is meaningless (a specific motion requires a specific amount of force). However, since for a redundant manipulator there exist multiple joint configurations for each task space configuration (provided the manipulator is not near a singular configuration). Hence it provides a possibility to include energy-saving optimisation in the null-space of the robot. Research in this direction can further improve the current framework. However, such optimisation will also demand the kinematic knowledge of the manipulator used and design of a task-specific cost function.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[3] Pankaj K Agarwal, J-C Latombe, Rajeev Motwani, and Prabhakar Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3124–3129. IEEE, 1997.

[4] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[5] Waseem Ahmad. Incremental learning of gaussian mixture models. 2010.

[6] Arash Ajoudani, Nikos Tsagarakis, and Antonio Bicchi. Tele-impedance: Teleoperation with impedance regulation using a body–machine interface. *The International Journal of Robotics Research*, 31(13):1642–1656, 2012.

[7] Srinivas Akella and Matthew T Mason. Posing polygonal objects in the plane by pushing. *The International Journal of Robotics Research*, 17(1):70–88, 1998.

[8] Nicolas Torres Alberto, Michael Mistry, and Freek Stulp. Computed torque control with variable gains through gaussian process regression. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 212–217. IEEE, 2014.

[9] Ermano Arruda, Michael J Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L Wyatt. Uncertainty averse pushing with model predictive path integral control. pages 497–502, Nov 2017. doi: 10.1109/HUMANOIDS.2017.8246918.

[10] Haruhiko Asada. Representation and learning of nonlinear compliance using neural nets. *IEEE Transactions on Robotics and Automation*, 9(6):863–867, 1993.

[11] Maria Bauza and Alberto Rodriguez. A probabilistic data-driven model for planar pushing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015. IEEE, 2017.

[12] Michael Beetz, Freek Stulp, Piotr Esden-Tempski, Andreas Fedrizzi, Ulrich Klank, Ingo Kresse, Alexis Maldonado, and Federico Ruiz. Generality and legibility in mobile manipulation. *Autonomous Robots*, 28(1):21, 2010.

[13] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[14] Dominik Belter, Marek Kopicki, Sebastian Zurek, and Jeremy Wyatt. Kinematically optimised predictions of object motion. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4422–4427. IEEE, 2014.

[15] Christopher M. Bishop. Mixture density networks. Technical report, 1994.

[16] Byron Boots, Arunkumar Byravan, and Dieter Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4021–4028. IEEE, 2014.

[17] David Braun, Matthew Howard, and Sethu Vijayakumar. Optimal variable stiffness control: formulation and application to explosive movement tasks. *Autonomous Robots*, 33(3):237–253, 2012.

[18] David J Braun, Florian Petit, Felix Huber, Sami Haddadin, Patrick Van Der Smagt, Alin Albu-Schäffer, and Sethu Vijayakumar. Robots driven by compliant actuators: Optimal control under actuation constraints. *IEEE Transactions on Robotics*, 29(5): 1085–1101, 2013.

[19] Jonas Buchli, Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Learning variable impedance control. *The International Journal of Robotics Research*, 30(7):820–833, 2011.

[20] E Burdet and Marnix Nuttin. Learning complex tasks using a stepwise approach. *Journal of Intelligent and Robotic Systems*, 24(1):43–68, 1999.

[21] Etienne Burdet, Rieko Osu, David W Franklin, Theodore E Milner, and Mitsuo Kawato. The central nervous system stabilizes unstable dynamics by learning optimal impedance. *Nature*, 414(6862):446, 2001.

[22] Sylvain Calinon, Irene Sardellitti, and Darwin G Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 249–254. IEEE, 2010.

[23] Sylvain Calinon, Petar Kormushev, and Darwin G Caldwell. Compliant skills acquisition and multi-optima policy search with em-based reinforcement learning. *Robotics and Autonomous Systems*, 61(4):369–379, 2013.

[24] Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344. IEEE, 2014.

[25] Olivier Cappé. Online expectation-maximisation. *Mixtures: Estimation and applications*, pages 31–53, 2011.

[26] David J Cappelleri, Jonathan Fink, Barry Mukundakrishnan, Vijay Kumar, and Jeffrey C Trinkle. Designing open-loop plans for planar micro-manipulation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 637–642. IEEE, 2006.

[27] Anindya Chatterjee. On the realism of complementarity conditions in rigid body collisions. *Nonlinear Dynamics*, 20(2):159–168, 1999.

[28] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. *CoRR*, abs/1610.00529, 2016. URL http://arxiv.org/abs/1610.00529.

[29] Stefano Chiaverini and Lorenzo Sciavicco. The parallel approach to force/position control of robotic manipulators. *IEEE Transactions on Robotics and Automation*, 9 (4):361–373, 1993.

[30] Akansel Cosgun, Tucker Hermans, Victor Emeli, and Mike Stilman. Push planning for object placement on cluttered table surfaces. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4627–4632. IEEE, 2011.

[31] Bruno Da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. *arXiv preprint arXiv:1206.6398*, 2012.

[32] Mark De Berg and Dirk HP Gerrits. Computing push plans for disk-shaped robots. *International Journal of Computational Geometry & Applications*, 23(01):29–48, 2013.

[33] Anthony Dearden, Yiannis Demiris, LP Kaelbling, and A Saffotti. Learning forward models for robots. IJCAI-INT JOINT CONF ARTIF INTELL, 2005.

[34] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[35] Yiannis Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–158, 2007.

[36] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5): 361–369, 2006.

[37] Yiannis Demiris and Andrew Meltzoff. The robot in the crib: A developmental analysis of imitation skills in infants and robots. *Infant and Child Development: An International Journal of Research and Practice*, 17(1):43–53, 2008.

[38] Miha Deniša, Andrej Gams, Aleš Ude, and Tadej Petrič. Learning compliant movement primitives through demonstration and statistical generalization. *IEEE/ASME transactions on mechatronics*, 21(5):2581–2594, 2015.

[39] Haris Dindo, Daniele Zambuto, and Giovanni Pezzulo. Motor simulation via coupled internal models using sequential monte carlo. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[40] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems VII*, 1, 2011.

[41] Finale Doshi-Velez. *Bayesian nonparametric approaches for reinforcement learning in partially observable domains*. PhD thesis, Massachusetts Institute of Technology, 2012.

[42] Jinjun Duan, Yahui Gan, Ming Chen, and Xianzhong Dai. Adaptive variable impedance control for dynamic contact force tracking in uncertain environment. *Robotics and Autonomous Systems*, 102:54–65, 2018.

[43] Zachary T Dydek, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control and the nasa x-15-3 flight revisited. *IEEE Control Systems Magazine*, 30(3):32–48, 2010.

[44] P Engel and M Heinen. Incremental learning of multivariate gaussian mixture models. In *BSAI*. Springer, 2010.

[45] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292, 2008.

[46] Yongxiang Fan, Wei Gao, Wenjie Chen, and Masayoshi Tomizuka. Real-time finger gaits planning for dexterous manipulation. *IFAC-PapersOnLine*, 50(1):12765–12772, 2017.

[47] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[48] Roy Featherstone and Oussama Khatib. Load independence of the dynamically consistent inverse of the jacobian matrix. *The International Journal of Robotics Research*, 16(2):168–170, 1997.

[49] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2016.

[50] J Randall Flanagan, Philipp Vetter, Roland S Johansson, and Daniel M Wolpert. Prediction precedes control in motor learning. *Current Biology*, 13(2):146–150, 2003.

[51] J Randall Flanagan, Miles C Bowman, and Roland S Johansson. Control strategies in object manipulation tasks. *Current opinion in neurobiology*, 16(6):650–659, 2006.

[52] David W Franklin, Etienne Burdet, Keng Peng Tee, Rieko Osu, Chee-Meng Chew, Theodore E Milner, and Mitsuo Kawato. Cns learns stable, accurate, and efficient movements using a simple algorithm. *Journal of neuroscience*, 28(44):11165–11173, 2008.

[53] Manan Gandhi, Yunpeng Pan, and Evangelos Theodorou. Pseudospectral model predictive control under partially learned dynamics. *arXiv preprint arXiv:1702.04800*, 2017.

[54] Gowrishankar Ganesh, Nathanael Jarrassé, Sami Haddadin, Alin Albu-Schaeffer, and Etienne Burdet. A versatile biomimetic controller for contact tooling and haptic exploration. In *2012 IEEE International Conference on Robotics and Automation*, pages 3329–3334. IEEE, 2012.

[55] Manolo Garabini, Andrea Passaglia, Felipe Belo, Paolo Salaris, and Antonio Bicchi. Optimality principles in variable stiffness control: The vsa hammer. In *2011 Ieee/Rsj International Conference on Intelligent Robots and Systems*, pages 3770–3775. IEEE, 2011.

[56] Zoubin Ghahramani. Probabilistic modelling, machine learning, and the information revolution. *presentation at MIT CSAIL*, 2012.

[57] Hiroaki Gomi and Rieko Osu. Task-dependent viscoelasticity of human multijoint arm and its spatial characteristics for interaction with environments. *Journal of neuroscience*, 18(21):8965–8978, 1998.

[58] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

[59] Sami Haddadin, Michael Weis, Sebastian Wolf, and Alin Albu-Schäffer. Optimal control for maximizing link velocity of robotic variable stiffness joints. *IFAC Proceedings Volumes*, 44(1):6863–6871, 2011.

[60] Open Source Haptics. *Moog Haptic Master Manual*. Moog FCS Robotics, 2008 (accessed May 9, 2019). URL https://www.h3dapi.org/modules/mediawiki/index.php/MOOG_FCS_HapticMaster.

[61] Masahiko Haruno, Daniel M Wolpert, and Mitsuo Kawato. Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10):2201–2220, 2001.

[62] Neville Hogan. Impedance control: An approach to manipulation. In *American Control Conference, 1984*, pages 304–313. IEEE, 1984.

[63] Matthew Howard, David J Braun, and Sethu Vijayakumar. Transferring human impedance behavior to heterogeneous variable impedance actuators. *IEEE Transactions on Robotics*, 29(4):847–862, 2013.

[64] Ping Hsu, John Mauser, and Shankar Sastry. Dynamic control of redundant manipulators. *Journal of Robotic Systems*, 6(2):133–148, 1989.

[65] Bidan Huang, Miao Li, Ravin Luis De Souza, Joanna J Bryson, and Aude Billard. A modular approach to learning manipulation strategies from human demonstration. *Autonomous Robots*, 40(5):903–927, 2016.

[66] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1398–1403. IEEE, 2002.

[67] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[68] Roland S Johansson and Kelly J Cole. Sensory-motor coordination during grasping and manipulative actions. *Current opinion in neurobiology*, 2(6):815–823, 1992.

[69] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258, 2011.

[70] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE, 2011.

[71] Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, and Stefan Schaal. Learning force control policies for compliant manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4639–4644. IEEE, 2011.

[72] Hilbert J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011, 2005.

[73] Hilbert J Kappen. Optimal control theory and the linear bellman equation. 2011.

[74] M Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, (6):718–727, 1999.

[75] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[76] Jennifer E King. *Robust Rearrangement Planning Using Nonprehensile Interaction*. PhD thesis, 2016.

[77] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[78] Marek Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Mörwald, and Jeremy Wyatt. Learning to predict how rigid objects behave under simple manipulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 5722–5729. IEEE, 2011.

[79] Marek Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Moerwald, and Jeremy L Wyatt. Learning modular and transferable forward models of the motions of push manipulated objects. *Autonomous Robots*, 41(5):1061–1082, 2017.

[80] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011.

[81] Klas Kronander and Aude Billard. Learning compliant manipulation through kinesthetic and tactile human-robot interaction. *IEEE transactions on haptics*, 7(3):367–380, 2014.

[82] Klas Jonas Alfred Kronander. Control and learning of compliant manipulation skills. 2015.

[83] Emo Kumar, Vikash; Todorov and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *ICRA*, pages 1071–1079, 2016.

[84] A.G. Kupcsik, M.P. Deisenroth, J. Peters, L. Ai Poh, V. Vadakkepat, and G. Neumann. Model-based contextual policy search for data-efficient generalization of robot skills. pages 415–439, 2017. URL http://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Kupcsik_AIJ_2015.pdf.

[85] Andras Kupcsik, Marc Peter Deisenroth, Jan Peters, Ai Poh Loh, Prahlad Vadakkepat, and Gerhard Neumann. Model-based contextual policy search for data-efficient generalization of robot skills. *Artificial Intelligence*, 247:415–439, 2017.

[86] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, Gerhard Neumann, et al. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 1401–1407, 2013.

[87] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[88] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[89] Alex X Lee, Henry Lu, Abhishek Gupta, Sergey Levine, and Pieter Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 177–184. IEEE, 2015.

[90] Gilwoo Lee, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Hierarchical planning for multi-contact non-prehensile manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 264–271. IEEE, 2015.

[91] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML-2013*, pages 1–9, 2013.

[92] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML (3)*, pages 1–9, 2013.

[93] Chao Li, Zhi Zhang, Guihua Xia, Xinru Xie, and Qidan Zhu. Efficient force control learning system for industrial robots based on variable impedance control. *Sensors*, 18(8):2539, 2018.

[94] Miao Li, Hang Yin, Kenji Tahara, and Aude Billard. Learning object-level impedance control for robust grasping and dexterous manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6784–6791. IEEE, 2014.

[95] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[96] Alain Liegeois. Automatic supervisory control of the configuration and behaviour of multibody mechanisms. *IEEE Transactions on systems, man and cybernetics*, 7(12): 868–871, 1977.

[97] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[98] Zongli Lin and Z Lin. *Low gain feedback*. Springer London, 1999.

[99] C Karen Liu. Dextrous manipulation from a grasping pose. In *ACM Transactions on Graphics (TOG)*, volume 28, page 59. ACM, 2009.

[100] Kevin M Lynch. The mechanics of fine manipulation by pushing. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2269–2276. IEEE, 1992.

[101] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[102] Jerry B Marion. *Classical dynamics of particles and systems*. Academic Press, 2013.

[103] M. T. Mason. *Manipulator grasping and pushing operations*. PhD thesis, MIT, 1982.

[104] Matthew T. Mason. Compliant sliding of a block along a wall. In *Experimental Robotics I, The First International Symposium, Montréal, Canada, June 19-21, 1989*, pages 568–578, 1989. doi: 10.1007/BFb0042542. URL https://doi.org/10.1007/BFb0042542.

[105] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001.

[106] Matthew Thomas Mason. Manipulator grasping and pushing operations. 1982.

[107] Michael J Mathew, Saif Sidhik, Mohan Sridharan, Akinobu Hayashi, Morteza Azad, and Jeremy L Wyatt. Online learning of feed-forward models for task-space variable impedance control. pages 597–605, Nov 2019.

[108] Michael J Mathew, Saif Sidhik, Mohan Sridharan, Akinobu Hayashi, Morteza Azad, and Jeremy L Wyatt. Online learning of feed-forward models for variable impedance control in manipulation tasks. Jul 2019.

[109] Jérôme Maye, Hannes Sommer, Gabriel Agamennoni, Roland Siegwart, and Paul Furgale. Online self-calibration for robotic systems. *The International Journal of Robotics Research*, 35(4):357–380, 2016.

[110] Hirokazu Mayeda and Y Wakatsuki. Strategies for pushing a 3d block along a wall. In *Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91*, pages 461–466. IEEE, 1991.

[111] José Ramón Medina, Dominik Sieber, and Sandra Hirche. Risk-sensitive interaction control in uncertain manipulation tasks. In *2013 IEEE International Conference on Robotics and Automation*, pages 502–507. IEEE, 2013.

[112] José Ramón Medina, Tamara Lorenz, and Sandra Hirche. Synthesizing anticipatory haptic assistance considering human behavior uncertainty. *IEEE Transactions on Robotics*, 31(1):180–190, 2015.

[113] Biren Mehta and Stefan Schaal. Forward models in visuomotor control. *Journal of Neurophysiology*, 88(2):942–953, 2002.

[114] R Christopher Miall and Daniel M Wolpert. Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279, 1996.

[115] Michael Mistry, Peyman Mohajerian, and Stefan Schaal. Arm movement experiments with joint space force fields using an exoskeleton robot. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 408–413. IEEE, 2005.

[116] Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar. Learning impedance control of antagonistic systems based on stochastic optimization principles. *The International Journal of Robotics Research*, 30(5):556–573, 2011.

[117] Kiyokazu Miyazawa, Yusuke Maeda, and Tamio Arai. Planning of graspless manipulation based on rapidly-exploring random trees. In *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.*, pages 7–12. IEEE, 2005.

[118] Yoshihiko Nakamura and Hideo Hanafusa. Optimal redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(1):32–42, 1987.

[119] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[120] Sundar Narasimhan. *Task level strategies for robots*. PhD thesis, Massachusetts Institute of Technology, 1994.

[121] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[122] Dennis Nieuwenhuisen, A Frank van der Stappen, and Mark H Overmars. Path planning for pushing a disk using compliance. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 714–720. IEEE, 2005.

[123] Ben North and Andrew Blake. Learning dynamical models using expectation-maximisation. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 384–389. IEEE, 1998.

[124] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018. URL http://arxiv.org/abs/1811.03378.

[125] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, may 2011. doi: 10.1109/icra.2011.5979561. URL https://doi.org/10.1109%2Ficra.2011.5979561.

[126] Rieko Osu and Hiroaki Gomi. Multijoint muscle regulation mechanisms examined by measured human arm stiffness and emg signals. *Journal of neurophysiology*, 81(4): 1458–1468, 1999.

[127] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[128] Michael A Peshkin and Arthur C Sanderson. The motion of a pushed, sliding workpiece. *IEEE Journal on Robotics and Automation*, 4(6):569–598, 1988.

[129] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2161–2168. IEEE, 2017.

[130] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[131] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[132] *Baxter, User Guide*. Rethink Robotics, 2014. URL http://mfg.rethinkrobotics.com/wiki/a/images/8/8c/Baxter_User_Guide_3.3.pdf.

[133] *Sawyer, User Guide*. Rethink Robotics, 2018. URL http://mfg.rethinkrobotics.com/wiki/a/images/1/1a/Sawyer_User_Guide_3.3.pdf.

[134] David A Rosenbaum. *Human motor control*. Academic press, 2009.

[135] Leonel Dario Rozo, Sylvain Calinon, Darwin Caldwell, Pablo Jiménez, and Carme Torras. Learning collaborative impedance-based robot behaviors. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[136] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[137] J Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, volume 19, pages 95–100. IEEE, 1980.

[138] Elliot Saltzman and JA Kelso. Skilled actions: A task-dynamic approach. *Psychological review*, 94(1):84, 1987.

[139] Stefan Schaal and Nicolas Schweighofer. Computational motor control in humans and robots. *Current opinion in neurobiology*, 15(6):675–682, 2005.

[140] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[141] Stanley A Schneider and Robert H Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. *IEEE Transactions on Robotics and Automation*, 8(3):383–394, 1992.

[142] John P Scholz and Gregor Schöner. The uncontrolled manifold concept: identifying control variables for a functional task. *Experimental brain research*, 126(3):289–306, 1999.

[143] Luc PJ Selen, David W Franklin, and Daniel M Wolpert. Impedance control reduces instability that arises from motor noise. *Journal of neuroscience*, 29(40):12606–12616, 2009.

[144] Kei Senda. Quasioptimal control of space redundant manipulators. In *Guidance, Navigation, and Control Conference and Exhibit*, page 4303, 1999.

[145] Reza Shadmehr and John W Krakauer. A computational neuroanatomy for motor control. *Experimental brain research*, 185(3):359–381, 2008.

[146] Aaron P Shon, Joshua J Storz, and Rajesh PN Rao. Towards a real-time bayesian imitation system for a humanoid robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2847–2852. IEEE, 2007.

[147] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[148] M. Song and H. Wang. Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering. In K. L. Priddy, editor, *SPIE Conference Series*, volume 5803 of *SPIE Conference-2005*, pages 174–183, March 2005. doi: 10.1117/12.601724.

[149] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 2012.

[150] Mike Stilman and James J. Kuffner. Planning among movable obstacles with artificial constraints. *International Journal of Robotics Research*, 27(12):1296–1307, 2008.

[151] Jochen Stüber, Claudio Zito, and Rustam Stolkin. Let's push things forward: A survey on robot pushing. *arXiv preprint arXiv:1905.05138*, 2019.

[152] Freek Stulp and Michael Beetz. Refining the execution of abstract actions with learned action models. *Journal of Artificial Intelligence Research*, 32:487–523, 2008.

[153] Freek Stulp, Evangelos Theodorou, Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning motion primitive goals for robust manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 325–331. IEEE, 2011.

[154] Hsi Guang Sung. *Gaussian mixture regression and classification*. PhD thesis, Rice University, 2004.

[155] CD Takahashi, Robert A Scheidt, and DJ Reinkensmeyer. Impedance control and internal model formation when reaching in a randomly varying dynamical environment. *Journal of neurophysiology*, 86(2):1047–1051, 2001.

[156] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, December 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953033.

[157] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.

[158] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

[159] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey Levine. Manipulation by feel: Touch-based control with deep predictive models. *arXiv preprint arXiv:1903.04128*, 2019.

[160] Emanuel Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural computation*, 17(5):1084–1108, 2005.

[161] Emanuel Todorov and Michael I Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002.

[162] H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2015. URL http://www.ausy.tu-darmstadt.de/uploads/Site/EditPublication/HoofHumanoids2015.pdf.

[163] Tie Wang, Goran S Dordevic, and Reza Shadmehr. Learning the dynamics of reaching movements results in the modification of arm impedance and long-latency perturbation responses. *Biological cybernetics*, 85(6):437–448, 2001.

[164] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016. doi: 10.1109/ICRA.2016.7487277.

[165] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[166] Thomas Wimbock, Christian Ott, and Gerd Hirzinger. Impedance behaviors for two-handed manipulation: Design and experiments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4182–4189. IEEE, 2007.

[167] Thomas Wimböck, Christian Ott, Alin Albu-Schäffer, and Gerd Hirzinger. Comparison of object-level grasp controllers for dynamic dexterous manipulation. *The International Journal of Robotics Research*, 31(1):3–23, 2012.

[168] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880, 1995.

[169] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

[170] Chenguang Yang, Gowrishankar Ganesh, Sami Haddadin, Sven Parusel, Alin Albu-Schaeffer, and Etienne Burdet. Human-like adaptation of force and impedance in stable and unstable interactions. *IEEE transactions on robotics*, 27(5):918–930, 2011.

[171] Tansel Yucelen and Wassim M Haddad. Low-frequency learning and fast adaptation in model reference adaptive control. *IEEE Transactions on Automatic Control*, 58(4): 1080–1085, 2012.

[172] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016.

[173] Jiaji Zhou, J Andrew Bagnell, and Matthew T Mason. A fast stochastic contact model for planar pushing and grasping: Theory and experimental validation. *arXiv preprint arXiv:1705.10664*, 2017.

[174] Claudio Zito, R Stolkin, Marek Kopicki, and Jeremy L Wyatt. Two-level RRT planning for robotic push manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS 2012)*, pages 678–685. IEEE, 2012.

[175] Claudio Zito, Rustam Stolkin, Marek Kopicki, and Jeremy L Wyatt. Two-level rrt planning for robotic push manipulation. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 678–685. IEEE, 2012.

Appendix A

Task space control law formulation

The problem task space control is formulated using rigid body assumptions starting with the manipulator equation. All notations in bold denote multi-dimensional quantities (i.e. matrices, vectors).

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \tau - \mathbf{J}_C^T \mathbf{F}_{ext}^i \tag{A.1}$$

where $\mathbf{q}$ is the (generalised) joint angle vector, $\dot{\mathbf{q}}$ is the joint velocity vector, $\ddot{\mathbf{q}}$ is the joint acceleration vector, $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis vector, $\mathbf{g}(\mathbf{q})$ is the gravity vector, and τ is the joint torque vector. $\mathbf{J}$ is the contact Jacobian and $\mathbf{F}_{ext}^i$ is the external force in the Cartesian space at step i. Here we are assuming $\mathbf{J}_C = \mathbf{J}$ the end effector jacobian. For ease of notational simplicity we write the same equation as:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \tau - \mathbf{J}_C^T \mathbf{F}_{ext}^i \tag{A.2}$$

where, $\mathbf{h} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})$. This equation is defined in the joint space of the robot and needs to be translated to its equivalent in the task space for implementing task space control.

The aim is to develop a controller which has a linear relationship between input (control input) and output (joint state acceleration of the robot). In other words, we like to make the entire robot behave like a spring either in the joint space (if a joint space controller is implemented) or in the task space (if a task space controller is implemented). Suppose we have a torque command as follows:

$$\tau = \mathbf{M}\mathbf{u} + \mathbf{h} \tag{A.3}$$

where, $\mathbf{u}$ is the control input to the robot and other parameters as described previously.

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \mathbf{M}\mathbf{u} + \mathbf{h} - \mathbf{J}^T\mathbf{F}^i_{ext} \tag{A.4}$$

$$\ddot{\mathbf{q}} = \mathbf{u} - \mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext} \tag{A.5}$$

$$\mathbf{J}\ddot{\mathbf{q}} = \mathbf{J}(\mathbf{u} - \mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext}) \tag{A.6}$$

$$\mathbf{J}\ddot{\mathbf{q}} = \mathbf{J}\mathbf{u} - \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext} \tag{A.7}$$

Recall from the fundamental relationship.

$$\mathbf{J}\dot{\mathbf{q}} = \dot{\mathbf{x}} \tag{A.8}$$

$$\mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} = \ddot{\mathbf{x}} \tag{A.9}$$

$$\mathbf{J}\ddot{\mathbf{q}} = \ddot{\mathbf{x}} - \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{A.10}$$

Substitute A.10 in A.7.

$$\ddot{\mathbf{x}} - \dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}\mathbf{u} - \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext} \tag{A.11}$$

$$\ddot{\mathbf{x}} = \mathbf{J}\mathbf{u} - \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext} + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{A.12}$$

Suppose we choose:

$$\mathbf{u} = \mathbf{J}^{-1}(\ddot{\mathbf{x}}_\mathbf{d} + {\mathbf{M}_\mathbf{d}}^{-1}\mathbf{K_P}(\mathbf{x_d} - \mathbf{x}) + {\mathbf{M}_\mathbf{d}}^{-1}\mathbf{K_D}(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}^i_{ext} \tag{A.13}$$

where, $\mathbf{M_d}$ is the desired inertia matrix and $\mathbf{x_d}$, $\dot{\mathbf{x}}_\mathbf{d}$, $\ddot{\mathbf{x}}_\mathbf{d}$ is desired components of the trajectory to be tracked. The term $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T = \mathbf{M}^{-1}_{ee}$ is the task space inertia of the robot. Since we are not interested in inertia shaping, we can keep the desired inertia matrix as the default end-effector inertia; that is $\mathbf{M_d} = \mathbf{M}_{ee}$, A.13 becomes, Now let us choose:

$$\mathbf{u} = \mathbf{J}^{-1}(\ddot{\mathbf{x}}_\mathbf{d} + \mathbf{M}^{-1}_{ee}\mathbf{K_P}(\mathbf{x_d} - \mathbf{x}) + \mathbf{M}^{-1}_{ee}\mathbf{K_D}(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{M}^{-1}_{ee}\mathbf{F}^i_{ext} \tag{A.14}$$

Let us assume that we are able to express $\mathbf{F}^i_{ext}$ as a spring model:

$$\mathbf{F}^i_{ext} = \mathbf{K_P}^i(\mathbf{x_d} - \mathbf{x}) + \mathbf{K_D}^i(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) \tag{A.15}$$

Then, A.14 becomes:

$$\mathbf{u} = \mathbf{J}^{-1}(\ddot{\mathbf{x}}_\mathbf{d} + \mathbf{M}_{ee}^{-1}(\mathbf{K_P} + \mathbf{K_P}^i)(\mathbf{x_d} - \mathbf{x}) + \mathbf{M}_{ee}^{-1}(\mathbf{K_D} + \mathbf{K_D}^i)(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) - \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{A.16}$$

Substituting A.16 in A.12, we get:

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{J}^{-1}(\ddot{\mathbf{x}}_\mathbf{d} + \mathbf{M}_{ee}^{-1}(\mathbf{K_P} + \mathbf{K_P}^i)(\mathbf{x_d} - \mathbf{x}) + \mathbf{M}_{ee}^{-1}(\mathbf{K_D} + \mathbf{K_D}^i)(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}})) - \mathbf{M}_{ee}^{-1}\mathbf{F}^i_{ext} \tag{A.17}$$

$$0 = (\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_\mathbf{d}) + \mathbf{M}_{ee}^{-1}(\mathbf{K_P} + \mathbf{K_P}^i)(\mathbf{x_d} - \mathbf{x}) + \mathbf{M}_{ee}^{-1}(\mathbf{K_D} + \mathbf{K_D}^i)(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) - \mathbf{M}_{ee}^{-1}\mathbf{F}^i_{ext} \tag{A.18}$$

$$\mathbf{M}_{ee}(\ddot{\mathbf{x}}_\mathbf{d} - \ddot{\mathbf{x}}) + (\mathbf{K_P} + \mathbf{K_P}^i)(\mathbf{x_d} - \mathbf{x}) + (\mathbf{K_D} + \mathbf{K_D}^i)(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) = \mathbf{F}^i_{ext} \tag{A.19}$$

$$\tilde{\mathbf{x}} = \mathbf{x_d} - \mathbf{x}$$

$$\mathbf{M}_{ee}\ddot{\tilde{\mathbf{x}}} + (\mathbf{K_P} + \mathbf{K_P}^i)\tilde{\mathbf{x}} + (\mathbf{K_D} + \mathbf{K_D}^i)\dot{\tilde{\mathbf{x}}} = \mathbf{F}^i_{ext} \tag{A.20}$$

Learning problem

Suppose we set:

$$\mathbf{K_P}_{mean} = \frac{\mathbf{K_P}_{max} + \mathbf{K_P}_{min}}{2} \tag{A.21}$$

$$\mathbf{K_D}_{mean} = \frac{\mathbf{K_D}_{max} + \mathbf{K_D}_{min}}{2} \tag{A.22}$$

and choose $\mathbf{K_P}^i$ and $\mathbf{K_D}^i$ such that:

$$\begin{aligned} \mathbf{K_P}^i &= f_1(\hat{\mathbf{F}}^i_{ext}) \\ \mathbf{K_D}^i &= f_2(\hat{\mathbf{F}}^i_{ext}) \end{aligned} \tag{A.23}$$

where, $\hat{\mathbf{F}}^i_{ext}$ is predicted external force.

If we can express this function as a parametarised basis function (for example the Gaussian kernel), we can use some learning methods to learn the parameters.

$$f_1(\hat{\mathbf{F}}^i_{ext}) = Kernel(\mathbf{u_{w_P}}, \Sigma_{w_P}) \tag{A.24}$$

$$f_2(\hat{\mathbf{F}}^i_{ext}) = Kernel(\mathbf{u_{w_D}}, \Sigma_{w_D}) \tag{A.25}$$

The external force needs to be learned and has to be expressed as function of the state.

Appendix B

Brief review on hardware and software built

B.1 Hardware

This section covers the different robots (Baxter and Sawyer) used in this thesis and a small description of the haptic device used to emulate the porridge environment.

Baxter

Baxter is a dual-armed robot designed and manufactured by Rethink Robotics, USA (Ret [132]). It was primarily designed as a collaborative robot which can safely interact with humans. Each arm of the robot consists of seven degrees of freedom. The joints of the arm actuated using serial elastic motors and hence are passively compliant. The robot gives precise position control, velocity control but lacks torque control. Baxter provides interface via ROS and their open source SDK.

Sawyer

Sawyer is a single arm robot from Rethink Robotics, USA (Ret [133]). This robot also consists of serial elastic actuators and has seven degrees of freedom. The robot can be controlled via ROS and provides position, velocity and torque control modes. Sawyer provides built in force sensing capabilities and the passive compliance make it capable to operate safely with humans.

Moog Haptic Master

The HapticMaster is a haptic device designed and developed by Moog FCS Robotics (Haptics [60]). The device is primarily used for human robot interaction and psychophysics studies. It is a force controlled device and can render arbitrary stiffness, damping and forces with minimal friction. The device consists of a force sensor at its end-effector which can be used to measure human interaction forces.

Design of the non-linear spring

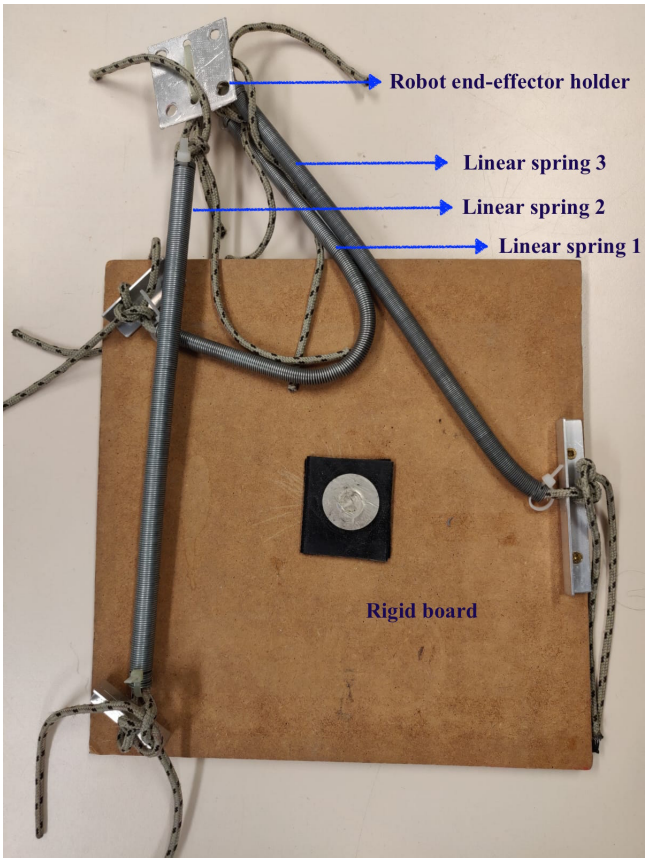


Figure B.1: Designed non-linear spring

The non-linear spring used consists of a combination of three linear springs of different (B.1) spring constants connected at an angle. This results in the setup pulling the robot end-effector with different forces in different directions. A plot of the spring extension in different directions against the force experienced is shown in Figure.

B.2 Software

This section covers a short description of the software stack developed as a part of the thesis as a collaborative effort between myself and Ermano Arruda for IRLab, University of Birmingham. [1].

AML:Advanced Manipulation and Learning (AML)

The stack is currently used by all research students at the IRLab for their work. Manipulating objects is one of the key domains actively researched in robotics and mainly comprises of dexterous manipulation (planning and changing contacts) and grasping. Initial methods in the domain were purely based on classical mechanics. Though these methods were successful in highly structured environments, they tend to fail in situations that included unmodeled scenarios like friction, variation in kinematic and dynamic properties of the objects etc. This inability to effectively and safely cope with uncertainty is still the bottleneck that inhibits the use of robots along with humans. Various methods in the machine learning domain were suggested to circumvent the problem of coping with this uncertainty. Apart from this, directly learning from data had the advantage of reduced complexity of manually modelling the scenario. One of the main challenges of using machine learning methods in manipulation (or robotics in general) is need of large quantities of data. This calls for methods to actively collect, store, check and retrieve the data as required. This software package puts forward some automated data collection infrastructure for manipulation. Apart from learning methods, usage of various trajectory optimisation methods gained prominence in manipulation. Trajectory optimisation methods tend to find an optimal trajectory (and or) policy for a task based on minimising a cost function. Also methods of directly encoding an human expert demonstration for a task execution were suggested in the past decade. Sometimes these demonstrations are also used to warm start the optimiser (this helps to converge to a good solution in less number of iterations). Having mentioned various methods of research in robot manipulation, it was our intention to bring several of these methods under a common software package. This package could serve as a common infrastructure (or starting point) for anyone in the IRLab or even as a open source project in the future. The stack includes many key methods actively researched like deep learning, classical methods, trajectory optimisation techniques and some data collection infrastructure for machine learning methods. This motivation led to the creation of the software package called Active Manipulation and Learning - AML. Apart from interfacing hardware platforms like Baxter, the package also supports 3D physics engines like Mujoco and Bullet and 2D engines

[1] https://github.com/RobotsLab/AML

like Box2D. The main advantage of the package is that any new hardware can be added to the existing stack by writing a single interface file. This makes the stack a general infrastructure for any amateur researcher in manipulation. In short, AML is a stack build for manipulation and learning for robot manipulators. The stack is written mostly in Python and partly in C++. It makes use of the ROS Indigo (Quigley et al. [130]) platform to create a general package for robot control, manipulation and grasping. The stack contains various packages that support classical control techniques, learning from demonstration, trajectory optimisation, deep learning and necessary support packages. The learning methods implemented depends on Google's tensorflow (Abadi et al. [1]).

Key points

- Highly modular.
- Interface with both hardware and software.
- Automatic manipulation data collection methods.
- Can be easily extended to any robotic platform.
- Availability of many recent algorithms, methods for manipulation.
- Actively maintained and tested.
- A good starting point for anyone in IRLab to pursue research on manipulation.
- A common library for the domain of manipulation.

Appendix C

Alternate attempted approaches

Before arriving at the proposed framework in Chapter 4, a set of studies were conducted to gain critical insight to some of the existing methods and understand the technical challenges involved in their implementations. These studies were successful in terms of understanding the approaches and their capability to solve different aspects of the thesis topic. The goal of the thesis was to make a planner capable of performing in-hand manipulation. Such a task involves creation of planner to plan contacts and a framework to apply optimal forces to manipulate the object.

During manipulation, stages might arise where the fingers will have to re-position the contact location due to instability of the object or their kinematic limits. The re-positioning of a finger from one contact location to another is called a finger gait. But such making and breaking of contacts results in discontinuous object dynamics. These discontinuities makes the problem of dexterous manipulation exceptionally hard.The change in dynamics can be smoothed by minimising the energy change of the object while initiating or leaving a contact. This can be achieved by making the fingers compliant while breaking and making contacts. Once the contacts are initiated, the fingers should be made stiff to effectively manipulate the object. Lack of stiffness after making the contact leads to slip and dynamic instability of the object.

A contact planner with all these capabilities is still an open problem in the literature. Solving the problem requires solving several constituent sub problems to be studied and solved. For instance, the dexterous manipulation requires to study dynamic discontinuities that arise during manipulation, variable impedance behaviour of the fingers to mitigate these discontinuities, representation of manipulation policies (skills), designing the search space on object surface to make contact etc. The following set of studies uses different type of popular approaches in the literature such as:

1 Deep Deterministic Policy Gradients for pushing, stiffness learning

2 QP based contact planner

3 Receding horizon manipulation planner

4 Movement Primitives(DMP & ProMP) based contact planner

5 PI2 based gait planner

6 Contextual Relative Entropy Policy Search (C-REPS)

7 Gaussian Process Relative Entropy Policy Search (GP-REPS)

The above mentioned seven implementations are to investigate these aspects. The main sub-questions investigated are:

1 Can a completely data driven approach learn an effective policy for dexterous manipulation? (Studies C, C)

2 What is the effective way to represent manipulation skills? (Studies C, C, C)

3 How to identify when to gait fingers? (Studies C, C, C)

4 How to find contact locations on an object surface? (Study C)

5 How to incorporate a learned dynamics model for policy search? (Study C)

6 How to anticipate a contact? (Study C)

7 Should a global or local forward-model be learned? (Study C)

8 How to vary stiffness while initiating contact and manipulating the object? (Study C)

Study 1: Learning ballistic compliant skills with DDPG

The main intention of this problem is to investigate whether a completely data driven approach is good for learning effective manipulation skills. The architecture starts from a randomly initialised policy and learns an optimal policy by collecting data from the robot and the environment.

Problem setting

The experiment is conducted in a physics engine (Bullet) based simulation platform. The environment consists of a box and a wall with a hole in it. A three link robot should learn a compliant policy to push the box inside the hole.

Approach

In this attempt, a version of the deep deterministic policy gradient (DDPG) was tested in simulation. DDPG is an off policy search method proposed in Lillicrap et al. [97]. An initial kinematic trajectory was given to the manipulator to push the box inside the hole. The algorithm aims to learn a set of state dependent torque commands to push the box inside the hole. The framework consists of an *actor-critic* neural network architecture. The *actor* consists of a 20 dimensional input vector and outputs a three dimensional joint torque value. The *critic* consists of the 23 dimensional input vector to output a one dimensional advantage factor. The input vector consisted of the joint position and velocity of the arm, end effector pose and position of the box in the end-effector frame. The reward function complimented for following the demonstration, for reaching the object and for pushing the object inside the hole.

Results

- The algorithm was able to learn a set of torque commands to push the box inside the hole[1].
- Further attempt to learn a state dependent stiffness value failed to produce good results.
- Attempting the same approach on robots with redundant degrees of freedom was also not successful[2].

Observations

- DDPG requires large amount of data to train.
- The algorithm learned a direct mapping between state and control commands instead of a policy distribution.
- The training time for the three link arm took 3 hours while the redundant manipulator took almost a day.
- This long time of training makes it hard to debug the approach effectively.
- The algorithm also had several hyper parameters with non-trivial values to be hand tuned.

[1] https://youtu.be/ee94RoWlCE4
[2] https://youtu.be/PzCVZ6ANkl0

- The convergence required around 700 epochs for training.
- The policy learned is task specific and hence not generalizable.
- The algorithm does not learn any explicit model of the task which can be reused.
- The structure of the neural network prevents from incorporating already known knowledge into the system.

Conclusion

DDPG learns a task specific global policy. The data complexity and training times for larger systems are practically impossible to satisfy with a real robot. This study showed various inabilities associated with a direct data based approach and motivated to use a model based approach.

Study 2: QP based contact planner

Quadratic Programming (QP) is popular optimisation technique used to solve constrained minimisation problems. This study aims to formulate the object manipulation problem as a quadratic minimisation problem, subject to different physics constraints.

Problem setting

The problem consists of an object initially grasped by a four finger manipulator. The task is to manipulate the object and gait the fingers as required.

Approach

This work was inspired by two works in the graphics literature by Fan et al. [46] and Liu [99]. In this approach, the gaiting problem is defined as a QP problem. The algorithm searches for a potential contact location in the neighbouring area of the current location subject to various physics constraints. The constraints include (i) finite search space around the current location (ii) manipulability of the finger and (iii) shape information of the object. While moving to the optimal contact location, the fingers slide on the object's surface resulting in manipulating it.

Results

- The algorithm successfully finds the adjacent contact locations.
- The contact locations found are within the manipulable range of the fingers.

- Attempts of the robot to slide to the new contact locations result in manipulating the object.

Observations

- The algorithm was only able to slide on the object's surface rather than making proper gaits [3].
- Finding a contact location far from the exiting location results in failure.
- Dynamic stability of the object is not considered.
- The search space had to be well defined in the neighbourhood of the existing contact location.
- The algorithm demands a smooth object surface.
- The method assumes point contacts with friction which is practically impossible to achieve with a physical robot.

Conclusion

The QP based approach is an intuitive approach to solve the problem of dexterous manipulation (with certain assumptions). The problem is to find better contact locations adjacent to current grasp locations subject to various constraints. By sliding along the shortest distance to newly found contact locations, forces are imparted on the object to manipulate it. The force applied at the contact locations are non-optimal. There is no notion of compliance while manipulating the object.

Study 3: Receding horizon manipulation planner

The success of the QP implementation motivated the following investigation. The same method was used to solve the problem of dexterous manipulation along with object dynamics.

Problem setting

The problem consists of an object initially grasped by a four finger manipulator. The task is to manipulate the object to a specified arbitrary configuration using an optimal sequence of commands.

[3] https://youtu.be/mMeV1oVj0dE

Approach

This work is inspired from the previous approach, yet is unique in several aspects. The algorithm is based on the idea of rigid body transformations. The approach assumes that the object is under a stable grasp using point contacts with friction. Using the forward-model of the object dynamics, a sequence of optimal wrenches to be applied on the object's centre of mass to reach a target pose can be computed, The resulting object trajectory from the optimal wrenches is computed by simulating the forward-model. The object trajectory contains potential poses which would be generated if the computed wrenches are applied at its centre of mass. Assuming no slip and with the knowledge of initial contact locations, a potential contact trajectory of the fingers on the object surface is computed using rigid body transformations. Once the contact trajectory is found, by using a polygonal approximation of the friction cone, a set of optimal forces to be applied at these contact locations can be obtained using quadratic programming. The friction cone approximation used in this work is similar to the one in Liu [99]. The computed contact trajectory and contact forces are used by a position-force controller in a receding horizon setting. In case of a slip, the algorithm re-computes the contact trajectory and object trajectory from the corresponding grasped location.

Results

- The algorithm was successful in manipulating the object to any arbitrary configuration[4].
- The computation time was negligible.

Observations

- Used a friction cone approximation for the contact forces.
- Assumes practically non-existent point contacts for problem solving.
- Failure to gait the fingers when their kinematic limits are reached.
- The approach demands full observability of the object state.
- The approach heavily relies on the accuracy of the kinematic information of both object and the fingers.
- Gaiting while manipulation is a non-trivial problem.

[4] https://youtu.be/4cNDvj0PPcU

Conclusion

This study gave insights to the analytic formulation of dexterous manipulation. The main shortcoming of this work is the assumption of contact type. The algorithm relies on stable initial contact of the object.

Study 4: Movement Primitives based contact planner

It is easier to use human demonstrations to initialise the optimiser or use them as a skill with an appropriate parameter based representation (Kumar and Levine [83]). This study aims at learning different techniques used to represent kinesthetic teaching of robots. One of the aim of this thesis is to use parametarised movement primitive (skill) to encode the demonstrations. Directly learning the modifying parameters of a skill is a much easier task compared to learning the entire skill from scratch. This study aims to learn about techniques which can represent manipulation skills from demonstration. The study also aims to find the right frame to represent the skills; i.e object frame or the robot frame or individual finger's tip frame.

Problem setting

The problem consists of an object initially grasped by a four finger manipulator. The aim is to make the manipulator follow a predefined set of skills specified in the object's frame.

Approach

The set of skills required to manipulate the object and gait the fingers are collected from demonstration. The skills are encoded using two popular techniques (i) Dynamic Movement Primitives (DMP)(Ijspeert et al. [66]) and (ii) Probabilistic Movement Primitives (ProMP)(Paraschos et al. [127]). The skills are followed using a position controller or an impedance controller. The skills were defined both in the robot frame and in the object frame. It was found that representing skills on object surface gives a pose invariant representation to the skills. Later, when a skill is to be followed, the object' surface skill is transformed to the manipulator frame. A finger can be either in manipulating or gait mode. The finger mode is determined by ranking fingers according to their manipulability. The least manipulable finger undergo gait first. Both manipulation primitives used in this study can scale temporarily and spatially. This helps to generalise the skills to objects of similar shape.

Results

- The implementation successfully manipulated the object using a set of pre-defined skills [5].
- The sequencing of the gait primitive after manipulating policy was successful.

Observations

- Defining skills on the surface of the object helps to preserve them irrespective of the pose of the object.
- By using an appropriate tracking controller, effective manipulation of the object can achieved.
- It is best to gait one finger at a time.
- The impedance controller was less stiff and resulted in slip from the intended skill.
- The approach lacked ability to independently choose gait target location.
- Modification of gait primitive under arbitrary object pose is non-trial due to collision.
- ProMPs does not have a notion of attractive goal state like DMP. This seems to cause instability at certain scaling conditions.

Conclusion

Effective manipulation skills can be collected from demonstration and are best defined in the object frame. Gait primitives are also defined in the object frame but modification of these skills under arbitrary object pose is non-trivial . The impedance controller alone is not capable to manipulate the object due to slip. Hence a variable impedance controller should be used. DMPs seems to be a better choice for movement primitive representation due to their inherent stability.

Study 5: PI2 based gait planner

One of the problems in manipulation is to reliably gait fingers as required. The movement primitives described in the previous section are capable of representing the gait primitive. But sometimes, the encoding parameters of the primitive have to be updated to completely

[5] https://youtu.be/37zzzK0ltac

modify the skill. This study investigates the use of a method proposed in Theodorou et al. [157] to address the issue of modifying primitive representation.

Problem setting

The problem consists of an object initially grasped by a four finger manipulator. The fingers are at their respective kinematic limits. A given gait primitive is to be modified to achieve a collision free path with the object.

Approach

The gait primitive is encoded in a suitable parametric form. The aim of this study is to find an effective way to modify the encoding parameters of the gait primitive to achieve a collision free path with the object. The collision of the default gait trajectory with the object can arise due to a particular pose of the object (typically reached during manipulation). The encoding parameters can be modified using a sampling based motion planner described in Arruda et al. [9]. The trajectory planner uses a PI2 based update to the parameters of the gait primitive to find a collision free path with the object. The update also takes into account various kinematic limitations of the manipulator and shape information of the object. Samples are generated around the given gait trajectory and are evaluated using a hand designed cost function. The cost function uses ray-casting to find collision of each point on the trajectory with the object. The trajectory is updated in the direction of low cost samples. Gradually the algorithm finds a path that is collision free with the object.

Results

- The implementation was able to modify a given gait primitive in collision with the object to be collision free. However, the approach failed to manipulate the object [6].

Observations

- Finding a collision free path with the object takes a lot of time to compute (in this attempt, around 25 seconds per finger to gait).
- Sometimes the optimiser fails to find a path due to the presence of a local minima in the cost function.

[6] https://youtu.be/MUPvgS2NbGU

- Defining the cost function demands complete knowledge of the object shape and the availability of a collision checker.
- The algorithm requires exact pose information of each fingers to perform precise collision checking.
- The approach do not use any information about the dynamics of the object.

Conclusion

The cost function design is non trivial and is not good for real time implementations due to the computational time. The study helped to understand the problems associated with the PI2 algorithm for the particular task of gait modification in manipulation.

Study 6: Contextual Relative Entropy Policy Search

A movement primitive can be considered as a lower level policy executed by the robot. Modifying the hyper-parameters can be considered as an upper level policy to the movement primitive. Policy search methods are effective in finding locally optimal parameters. The use of policy search for finding parameters of a lower level policy reduces the search space for the task. The main aim of the study described in this section is to find a method which can effectively learn the hyper-parameters of a skill. The algorithm used here was proposed in Kupcsik et al. [86] .

Problem setting

The task involves the classic peg insertion problem. A rectangular peg is attached to the end-effector of the Sawyer robot. The goal is to correctly insert the peg into a upward facing rectangular hole with 1.5mm of allowance. The orientation (with z axis) of the hole is set to randomly change. The robot has to learn a general policy applicable to different orientations.

Approach

The aim of this study is to investigate a policy search technique which can learn the hyper-parameters of a movement primitive. Consider the skill of *forward hand* in tennis. There will be many instances where the *forward hand* is the right skill to counter an incoming ball. But the player might also require to do minor adjustments to the same skill primitive in different cases of the same *forward hand* task. This scenario can be thought as follows. All the situations for which the *forward hand* is the right solution can be treated as a cluster.

Within this cluster, there will be several instances where the skill has to be modified to meet certain requirements (e.g. hitting to a particular location of the court). The points within a cluster are called as contexts. In other words, there will be a fixed skill or a chain of skills associated with a cluster of contexts. In each of these contexts, we may need to modify the hyper-parameters of the same skill according to the requirements.

The idea described here is inspired by the tennis playing scenario. Consider the peg insertion problem where the hole is facing upwards. A set of skills are required to push the peg inside the hole. Now consider the case of changing orientation of the hole (i.e rotation about the z axis). The robot could use the same skill (with appropriate modifications) for all the different cases of orientation change about z axis. The random orientation change can be thought as a context. The policy search described here aims to learn an upper level policy for all the contexts in a cluster (i,e, the skills are fixed for the cluster, we are only learning their hyper-parameters). In this problem, each orientation of the hole is considered as a context (s). The aim is to find a optimal policy $\pi(w|s)$, where w is the hyper-parameters of the skills. The skill is defined as a DMP. The aim of the policy search method is to come up with a set of parameters to modify the DMP to adapt it for the observed context.

The peg insertion problem is solved as a sequence of two DMPs. The first DMP is used to reach the hole and second DMP is used to insert the peg to the hole. The peg has to be exactly aligned with the hole to be successfully move inside it. In this problem setting, the w denotes the spatial scaling parameters of the DMP as well as the orientation of the peg. The policy search has to find these six parameters for different hole orientations (contexts). The cost function consists of three terms (i) alignment reward (make the peg aligned with the hole) (ii) closeness reward (get close to the hole to insert in) and (iii) insertion reward (move inside the hole as deep as possible). Once the set of parameters are found, new skill trajectory is computed and is followed by using a position controller.

Results

- The implementation was successful [7] in learning all the six parameters for hole orientation change from -5 to +5 degrees.

Observations

- The algorithm was able to find a solution in 200 epochs.
- The policy is updated using samples hence is susceptible to the bias in the samples.

[7] https://youtu.be/R_JtEG6Hxuk

- The policy search uses large number of samples which scales with increase in parameter dimension.

Conclusion

The success of the policy search method relies on the effective parameterisation of the skills. The method does not learn or approximate any model of the task, this results in larger number of trials on the robot.

Study 7: Gaussian Process Relative Entropy Policy Search

The problem with direct policy search methods is the lack of a model. Model learning involves collecting several examples from the real robot. Collecting data from the real robot is expensive. If the learning process can be augmented by a model of the system, it can be used to generate additional data about the task. The problem was attempted using a model based extension of C-REPS Kupcsik et al. [86] explained in Section C.

Problem setting

The task involves a classic peg insertion problem. A rectangular peg is attached to the end-effector of the robot. The goal is to correctly insert the peg into a rectangular hole with 1.5mm allowance. The orientation of the hole is set to randomly change within a range. The robot has to learn a general policy applicable in different orientations to complete the peg insertion.

Approach

The aim is to collect data and then learn a forward-model of the system. Later this model is used for policy search. Insights into the algorithm are described in Kupcsik et al. [86]. Learning of the forward-model helps in reducing the data complexity of the algorithm. The forward-model used consists of multiple models. One model is used to learn the contexts and another model is used to learn the reward function. Gaussian Process regression is used to learn the reward function. The contexts are learned using the maximum likelihood estimation of a Gaussian distribution. Later this learned model is used to generate additional data during the policy search operations. After a pre-defined number of iterations using the forward-model, additional data are collected from the real robot using the updated policy. These steps are repeated until convergence of the reward.

Results

- The implementation was successful in learning all the six parameters for a hole orientation change from -5 to +5 degrees.
- Same technique was also tried for context changes from -12 to +12 degrees. The algorithm successfully [8] learned for all angles between -12 and 0 degree while failed for the ones from 0 to 12 degrees.

Observations

- The algorithm was able to find an optimal solution in 70 epochs for the -5 to 5 degree case.
- The solution depended on the initial set of random samples. Hence collecting them in such a way as to span the search space results in faster convergence and better solutions.
- The forward-model learned can be a set of different models instead of a single global model. This allows incorporating more structure to the forward-model to learn faster.
- Each of the separate models can be learned using different function approximators.
- The use of different contexts for forward-model learning helps to choose models in accordance with the data instead of using a single global model.

Conclusion

The algorithm offers a model based reinforcement learning to find a higher level optimal policy that dictate the hyper parameters of skills to complete the peg insertion problem. This study showed that using a forward-model can greatly reduce the sample complexity of the algorithm. The algorithm uses two separate techniques to learn the context distribution and the reward distribution. This supports the use of multiple modular models for learning the forward-model of the task.

[8] https://youtu.be/rC3hVlMbH2Y