Incorporating Contextual Knowledge Into Human-Robot Collaborative Task Execution

Ahmed Faisal Abdelrahman

Publisher: Dean Prof. Dr. Wolfgang Heiden

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science

Sankt Augustin, Germany

March 2020

Technical Report 01-2020

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-080-3

This work was supervised by
Prof. Dr. Paul G. Plöger
M. Sc. Alex Mitrevski

Digital Object Identifier https://doi.org/10.18418/978-3-96043-080-3

Abstract

An essential measure of autonomy in service robots designed to assist humans is adaptivity to the various contexts of human-oriented tasks. These robots may have to frequently execute the same action, but subject to subtle variations in task parameters that determine optimal behaviour. Such actions are traditionally executed by robots using pre-determined, generic motions, but a better approach could utilize robot arm maneuverability to learn and execute different trajectories that work best in each context. In this project, we explore a robot skill acquisition procedure that allows incorporating contextual knowledge, adjusting executions according to context, and improvement through experience, as a step towards more adaptive service robots.

We propose an *apprenticeship learning* approach to achieving context-aware action generalisation on the task of robot-to-human object hand-over. The procedure combines learning from demonstration, with which a robot learns to imitate a demonstrator's execution of the task, and a reinforcement learning strategy, which enables subsequent experiential learning of contextualized policies, guided by information about context that is integrated into the learning process. By extending the initial, static hand-over policy to a contextually adaptive one, the robot derives and executes variants of the demonstrated action that most appropriately suit the current context. We use dynamic movement primitives (DMPs) as compact motion representations, and a model-based Contextual Relative Entropy Policy Search (C-REPS) algorithm for learning policies that can specify hand-over position, trajectory shape, and execution speed, conditioned on context variables. Policies are learned using simulated task executions, before transferring them to the robot and evaluating emergent behaviours.

We demonstrate the algorithm's ability to learn context-dependent hand-over positions, and new trajectories, guided by suitable reward functions, and show that the current DMP implementation limits learning context-dependent execution speeds. We additionally conduct a user study involving participants assuming different postures and receiving an object from the robot, which executes hand-overs by either exclusively imitating a demonstrated motion, or selecting hand-over positions based on learned contextual policies and adapting its motion accordingly. The results confirm the hypothesized improvements in the robot's perceived behaviour when it is context-aware and adaptive, and provide useful insights that can inform future developments.

Acknowledgements

I would like to begin by thanking Prof. Paul G. Plöger, to whom I am grateful for sparking my interest in several topics that were essential to my research during the year leading up to the completion of this report, and for supporting my pursuit of this project.

I would like to extend special thanks to my second supervisor Alex Mitrevski, for providing supportive guidance throughout the project and being a constant source of valuable advice, as well as introducing me to various intriguing facets of this research, without which this work would not have successfully materialized.

I would also like to express my deepest gratitude to my family, and to my parents in particular, whose immense support and endless encouragement never cease to be my primary source of motivation.

Finally, I would like to acknowledge the help provided by the participants of our user study, who had graciously volunteered to contribute to one of the main findings of this project.

Contents

List of Symbols

Symbol	Description
t	time
s_t or x_t	state at time-step t
a_t or u_t	action at time-step t
$\mathcal{A}$	set of applicable actions
$\mathcal{S}$	set of possible states
$\mathcal{R}$	reward function/model
$\mathcal{P}$ or $\mathcal{T}$	state-transition model
π	policy
π^*	optimal policy
$\pi(s_t)$	deterministic policy
$\pi(a_t \vert s_t)$	stochastic policy
r_t	immediate reward at time-step c context
τ	roll-out
G_t^τ	return of roll-out τ, from time-step t onwards
J_π	policy performance measure
γ	discount factor
N	number of policy iterations
M	number of roll-outs (per policy iteration)
$\mathbf{c}$	context vector
$\mathcal{C}$	context set
D_c	dimensionality of context vector
$\mu(c)$	context distribution
$\mathcal{N}(\omega \vert \mu(c), \Sigma)$	context-dependent linear-Gaussian model
$\mathbf{a}$	linear-Gaussian policy parameter
$\mathbf{A}$	linear-Gaussian policy parameter
$\mathbf{\Sigma}$	covariance matrix
θ	upper-level policy parameters
ω	lower-level policy parameters
D_ω	dimensionality of lower-level policy parameter
$p(c, \omega)$	trajectory distribution (REPS)
$\phi(c)$	context feature vector (REPS)
$p^{[i]}$	trajectory sample weight (REPS)
η and $\widetilde{\theta}$	Lagrangian parameters (REPS)
$\mathbf{g}$	DMP goal pose parameter
$\mathbf{w}$	DMP basis function weights (trajectory shape parameters)

$\mathbf{W}$	trajectory shape parameters vector
τ	DMP time-scale parameter
y	DMP system state (controlled variable)
x	DMP phase variable
f	DMP forcing term
N_{dmps}	DMP number of basis functions
N_{bfs}	DMP number of basis functions
ψ	DMP basis functions
$V(s)$	State-value function
$Q(s, a)$	Action-value function
ϵ	relative entropy bound
δ_t	TD error
κ	bounding box height-width ratio
$d(\cdot, \cdot)$	distance measure
z	force sensor readings

List of Abbreviations

Acronym	Full Form
APID	Approximate Policy Iteration with Demonstration
Black-DROPS	Black-box Data-efficient Robot Policy Search
CBA	Context-based Architecture
CECER	Covariance Estimation with Controlled Entropy Reduction
CK	Contextual Knowledge
CMB	Context-mediated Behaviour
C-MORE	Contextual Model-based Relative Entropy Stochastic Search
CPS	Contextual Policy Search
C-REPS	Contextual Relative Entropy Policy Search
CrKR	Cost-Regularized Kernel Regression
DDPG	Deep Deterministic Policy Gradients
DLJ	Dimensionless Jerk
DMP	Dynamic Movement/Motion Primitive
DOF	Degree of Freedom
DP	Dynamic Programming
DQfD	Deep Q-Learning from Demonstrations
DQN	Deep Q-Network
EM	Expectation-Maximization
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
GP	Gaussian Process
GPDMP	Global Parametric Dynamic Movement Primitive
GPR	Gaussian Process Regression
GPREPS	Gaussian Process Relative Entropy Policy Search
GPS	Guided Policy Search
HER	Hindsight Experience Replay
HiREPS	Hierarchical Relative Entropy Policy Search
HMM	Hidden Markov Model
HO	Hand-Over
HRI	Human-Robot Interaction
HRL	Hierarchical Reinforcement Learning
HROHO	Human-Robot Object Hand-Over
HSR	Human Support Robot
IF	Information Fusion
IRL	Inverse Reinforcement Learning

LfD	Learning from Demonstration
LWR	Locally Weighted Regression
MC	Monte Carlo
MDP	Markov Decision Process
MP	Movement/Motion Primitive
NAC	Natural Actor-Critic
NP-REPS	Non-parametric Relative Entropy Policy Search
PCA	Principal Component Analysis
PbD	Programming by Demonstration
PG	Policy Gradient
PI^2	Policy Improvement by Path Integrals
PILCO	Probabilistic Inference for Learning Control
PoWER	Policy learning by Weighting Exploration with Returns
ProMP	Probabilistic Movement/Motion Primitive
PS	Policy Search
PWTD	Point-wise Trajectory Differences
REPS	Relative Entropy Policy Search
RL	Reinforcement Learning
SAL	Spectral Arc Length
TD	Temporal Difference
WMLE	Weighted Maximum Likelihood Estimation

1

Introduction

In the near future, service robots deployed to autonomously interact with and assist people in commonplace tasks must possess degrees of adaptivity that exceed those of today's prototypical robots. The highly uncertain and dynamic environments they are expected to occupy with humans, learn from, and act intelligently in necessitate methods more advanced than conventional adaptation strategies, especially as our conception of robot intelligence, and thus expectations, evolve beyond simple reactive and goal-directed behaviours. The execution of common physical tasks, such as serving drinks or opening doors, can be significantly enhanced when made adaptive to and dependent on different operating conditions, as opposed to static, pre-defined actions, whose results may not satisfy variations of the same task.

In robotics research, complex human-inspired behaviours are often pursued by studying and extracting human tendencies and concepts in order to transfer them to robots: an approach we can take to work towards more intelligent robot adaptivity. The concept of context is a vital factor we humans take into consideration to optimize the things we say, the ways in which we act, and how we adapt best to the current situation. Despite being an inherently vague concept, it is difficult to deny that people's actions can have drastically varying outcomes in different contexts, and that mediating behaviour according to context is a natural human instinct. In much the same way, autonomous robots' actions should adhere to differences in contexts relating to their task, particularly when they involve interacting or collaborating with humans, which can be achieved by endowing them with similar capabilities of context awareness and adaptivity. These would constitute an additional dimension of a robot's decision making processes, and could potentially increase their robustness to variations in human-oriented tasks, in particular.

In this project, we propose utilizing some form of contextual knowledge to improve a robot's execution of a collaborative task, by adapting its activity to the current perceived context. We refer to context as some representation of the current state of the environment, task, and/or persons involved, which dictates the optimal manner in which the task can be performed. The aim is to represent an action such that it could be generalized to different contexts, and potentially improvable through experience. This capability is expected to equip a robot with an added degree of adaptivity, such that it produces more appropriate and efficient task executions according to a given situation.

To that end, we review the literature on context representations and various approaches to robot motor skill generalization in order to identify a viable method for learning context-aware behaviour in a chosen task. We then implement and evaluate the results attained under the proposed approach, through

quantitative and qualitative means. The primary contributions of this project are an implementation of a context-aware robot motor skill, and a user study conducted to confirm hypothesized improvements over a more conventional non-adaptive approach to task execution, in order to justify further pursuing the idea of incorporating contextual knowledge into a robot's human-oriented activities.

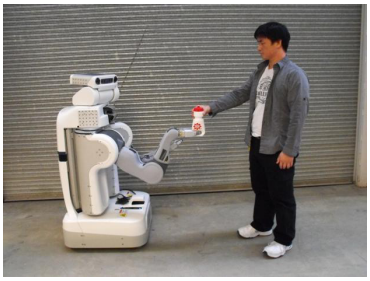
In this chapter, we introduce and formalize our problem, then present the solution devised in this project. Section 1.1 introduces the robot task we employ as a use case throughout the project, section 1.2 discusses the main motivation of the work, section 1.3 elaborates on a relevant challenge concerning the underlying dilemma of 'learning versus modelling', section 1.4 formulates the problem addressed in the project more precisely, and section 1.5 presents the approach we propose for addressing the problem.

1.1 A Representative Use Case: Robot-to-Human Object Hand-Over

Among the most rudimentary, and possibly the most essential, instances of a collaborative task is the act of handing on object over to a person, which is inarguably vital for everyday interactions and routine tasks. Object hand-overs are considered to be one of the elementary actions with which a robot can physically interact and cooperate with people, and even collaborate and assist in simple, repetitive tasks [7] [115]. Fairly early research highlighted the importance of hand-overs as a collaborative task, particularly in the exploration of human-robot coordination ([55]), and investigated human hand-overs to uncover underlying principles that could pave the way for deriving similar behaviours in robots ([98]). For these reasons, and its relative simplicity in concept and implementation, human-robot object hand-overs have been chosen as a suitable test bed for applying the notion of contextual adaptivity and studying its implications on robot behaviour and consequent user perceptions.

Intuitively, the problem of human-robot object hand-overs generally involves the appropriate exchange of some object, whether the robot is the giver or the receiver. Particularly in the context of robotics, the dyadic task is often segmented into discrete phases such as the approach, passing, and retraction phases ([74]), and thus modeled as a sequence of independent actions. In this work, we concentrate on robot-to-human hand-overs, and frame the problem as one in which a service robot must suitably present a held object to a person they have approached (Figure 1.1). More concretely, the focus of the task is on the movement of the object along some trajectory towards a chosen hand-over position, amounting to the 'passing phase'. It is this characteristic motion that affords rich measures of contextual adaptivity.

Traditionally, robot hand-overs are accomplished in a pre-defined motion that acceptably fulfils the objective of the task, often as a pre-planned trajectory or the output of a tracking controller that ensures the robot's end-effector reaches some hard-coded position, in each execution. Instead, a different approach could utilize the manoeuvrability of a robot's arm and the range of trajectories it can generate to more intelligently choose the way it executes the hand-over, given the knowledge that some trajectories work better in some contexts than others. Examples include placing the object lower for a seated person, or following a slower velocity profile to account for an object's fragility or a person's impairment. This approach to transforming the robot's hand-overs from a mere repetitive action to a contextualized motor skill constitutes the contextual adaptivity we seek to implement and evaluate in this project, with the



(a)

(b)

(c)

(d)

Figure 1.1: Examples of typical human-robot object hand-overs: a) Object exchange with a PR2 robot (CARIS, The University of British Columbia), b) Handing off an object to BERT-2 (Bristol Robotics Laboratory), c) Jido Robot Object Hand Over using a Human Aware Manipulation Planner (RoboticsLAAS), d) Accepting an object from El-E, a robot designed to aid users with mobility impairment with everyday tasks (Rob Felt/Georgia Tech University).

aforementioned examples demonstrating contextual knowledge that is instinctive to humans and would have to be incorporated in the robot's skill.

As a requisite for realizing some form context-aware hand-overs, it is important to consult the wealth of literature on inter-human and human-robot object hand-overs to gather some general insights. Due to the ubiquity of the task and interest in the complex mechanisms that enable humans to execute it with impressive ease, various researchers have conducted human studies to extract strategies applied and considerations taken by people as they perform hand-overs ([46][100][92][94]). Among the resulting findings are the multiple reasons for which humans adapt hand-overs, including preferences, mutual comfort, and mobility, and the fact that implemented robot hand-overs are best received by users when their parameters (pose, trajectory, etc.) are tuned to respect these factors. Other studies investigated human preferences for robot hand-overs more thoroughly, particularly how natural and appropriate different robot hand-over configurations are perceived ([20]), and preferred robot approach distances, hand-over positions, and so on ([58]). The results of these studies and several others collectively indicate users' preference for multi-faceted adaptivity, as opposed to lack thereof, which lends some credibility to the idea of robot

context-awareness being a worthwhile pursuit.

Our objective, then, is to construct a generalized robot hand-over skill, by providing a robot with some expression of contextual knowledge and the ability to dynamically change trajectory characteristics based on that information to adapt hand-overs to contexts. The virtue of this is then verified by evaluating whether it improves the robot's behaviour, as perceived by standard users.

1.2 The Importance of Adaptivity

The primary motivation of this work is exploring and validating an approach to achieving a human-inspired form of task adaptivity. The concept of adaptivity in behaviour is undoubtedly among the foremost requirements for the success of robots in unstructured real-world environments, whether it involves simple reconfiguration protocols, learning capabilities, or other adaptation strategies. As we focus on adaptivity of service robots in high-level task execution, particularly in association with humans, we may associate this desired form of adaptivity with the term *versatility*: the capability of satisfying various functions, or conditions of a task, competently. We postulate that adaptivity to varying task contexts and possessing versatile actions or, more appropriately, 'skills' whose characteristics can be adjusted to these contexts is an important capability for autonomous service robots.

The virtue of adaptive behaviours in robots designed to interact and collaborate with humans is increasingly apparent from the conclusions of various user studies. People perceive adaptive robots as being more competent and intelligent, as opposed to their non-adaptive counterparts, even evoking terms such as 'helpful', 'attentive', and 'courteous' in users asked to describe their behaviour, and achieving superior results in metrics including user experience and task performance when tested in domestic, workplace, and healthcare settings ([6][73][43]). Measures of adaptivity which elicit such positive perceptions in target users thus increase chances of the ubiquitous deployment of assistive robots, since they appeal to our instinctive preferences and inclinations. In cases of physical collaborations, user experience considerations are just as important as maximizing task performance in the course of designing robot assistants ([46]). Various studies on robot hand-overs, in particular, have highlighted user preferences for adaptive behaviours, mainly in the form of context-aware execution adjustments ([72]). Robot actions that do not conform to contextual differences risk seeming unnatural and unsatisfactory to users, in the best case scenario, or possibly failing to accomplish their objective, in the worst.

The field of human-robot collaboration (HRC) generally aims to attain robust robot execution of collaborative, human-oriented tasks; a goal that is primarily hindered by the unpredictable nature of humans. This substantial obstacle in the way of deploying robots that can work alongside humans has prompted attempts at human intent prediction and long-term learning strategies, as well as extensive studies of peoples' tendencies, to cope with variability in task specifics and people. The difficulty in dealing with this variability motivate designing adaptive robot behaviours and/or skills, such that robots are equipped to adapt and reasonably cope with it ([39]). Versatile skills are then expected to facilitate handling different peoples' attributes, such as physical traits, mobility, preferences, etc., and situational task contexts to produce more natural executions, as inspired by human competence.

Inflexible robot behaviour and actions can be inefficient for collaboration and joint tasks in particular,

due to such tasks usually being highly dynamic and significantly dependent on contextual differences. For example, the act of collaboratively carrying an object with a person requires adapting to their stature, movement gait, and any unorthodox movements, while the optimal way to hand an object over depends on the receiver's posture and capabilities, their preferences, and the object's characteristics. While humans are efficient at conforming to dynamic changes, it remains a significant challenge to contemporary robots, which do not generally possess an awareness of or adaptivity to context. Therefore, identifying and applying viable methods to achieve the intended adaptivity in robots is an issue this project aims to address, at least in part.

1.3 The Learning/Modeling Problem

As previously mentioned, this work is motivated by the importance of robot adaptivity to different contexts of a task, which introduces a significant challenge to service robots designed for operating around and with humans. These robots must be equipped to handle uncertainty in situations and attributes of people to cope with the dynamic real-world environments they are envisioned to operate in. Conventional approaches suggest adopting one of two ways to achieve such capabilities, both of which have limiting drawbacks, and together give rise to the familiar learning versus modeling dilemma.

The first direction involves employing traditional machine learning strategies to exclusively learn the best mode of action for every conceivable scenario. Example candidate methods include training deep neural networks in a supervised fashion with data indicating optimal behaviour in each context, such that the robot performs the best action given some input on the situation. Unfortunately, this would necessitate large amounts of training data, particularly pertaining to the robot's context-specific executions, and thus prohibitive amount of trials on the robot.

An orthogonal approach could be to construct explicit models of the action, incorporating the dynamics that drive the motions of the robot as it pursues the objective of its task. However, it is generally difficult to obtain accurate, unbiased dynamics models for high-dimensional problems, making it potentially more complicated than a learning approach. In addition, this difficulty would be exacerbated by possibly having to obtain adequate models that independently satisfy every mode of behaviour corresponding to a respective context, or constructing a single model that sub-optimally satisfies as many contexts as possible.

Neither extreme is suitable for handling all possible context-dependent variations of the same action, due to foreseeable tractability concerns, or unsatisfactory resultant behaviour. More promising approaches could involve combining the two methods or devising more novel solutions based on human tendencies, which this project aims to investigate. Considering that repetitive robot actions such as object hand-overs usually exhibit subtle task variations, a model of an exemplary execution can initially be learned or captured. Then, this would be utilized to subsequently construct a generalizable skill by equipping the robot with the ability to learn derivative variants of the execution that work best in each context, and thus tuning behaviour to observed contexts. Finding such a solution thus emerges as a primary challenge.

An ideal approach could obviate the need for either exclusive learning or comprehensive modelling,

both alleviating their respective limitations and approaching the kind of strategies us humans would apply. The latter consideration is important, since it is subsequently more likely to produce a measure of adaptivity that would appeal to users in general. As an example, more fluent robot object hand-overs would result from generalizing the motions required by the robot to different conditions, such as the posture of the receiver, as a normal person would, obviating the need for explicit programming of every possible scenario, and avoiding reliance on a single, non-adaptive motion.

1.4 Problem Statement

In this project, we aim to identify practical approaches to incorporating contextual knowledge into repetitive collaborative robot actions, and evaluating the applicability and efficacy of a concrete implementation. A robot may frequently be required to perform the same action (such as helping a person carry an object), but subject to subtle variations that affect the optimal execution of the task (such as the height and dexterity of the person). On the one hand, modeling or learning all possible scenarios for such an action is inconceivable, while static, context-independent actions risk being too rigid and unnatural for human standards. Skill representations that allow for adjustments according to varying contexts and improvement with experience are expected to increase the robustness of collaborative robots.

As a requisite to achieving context-aware behaviours, it is pertinent to provide a comprehensive overview of any existing works that tackle the same problem of contextual adaptivity, or suggest promising solutions. This would provide a clearer picture of the state-of-the-art in the addressed problem.

Having selected human-robot object hand-overs as an adequate use case on which to base our work, we must formulate the characteristics of the task and define contextual variations associated with it. The hand-over action is then implemented on a robot platform capable of accomplishing the task, such that subsequent evaluations of emergent behaviours are made on the basis of the robot's performance.

Generalizing the robot's hand-over skill to different contexts requires identifying an appropriate method or algorithm that enables it to utilize contextual knowledge to guide its execution of the task. In this case, the knowledge would constitute indications of the optimal hand-over positions or velocities, for example, that match context factors such as the posture of a person, their physical ability, the presence of an intermediate counter, etc., such that the controllable aspects of the motion are varied to respect these conditions. As previously alluded to, this is consistent with human behaviour: we change the way we perform this action after perceiving such differences, and the chosen method should produce a similar effect.

Ultimately, we aim to address the absence of a systematic evaluation of a context-adaptive approach on a real robot task involving collaborating with a human. Although promising solutions were initially identified, such as the contextual policy search approach of the GPREPS algorithm ([66]), they have not been, to the best of our knowledge, applied and tested in a real autonomous robot scenario. Therefore, we conduct a user study in which emergent, contextually adaptive behaviours applied to the hand-over task are compared to traditional, robot-centric executions. This study would involve experiments that would allow us to conclusively validate the improvements that the applied method brings to robot behaviour, as perceived by human users. This then provides empirical evidence either confirming or invalidating our

hypothesis concerning the expected benefits of contextual adaptivity to human-oriented task execution, and thus justify pursuing this idea further in the future.

1.5 Proposed Approach

We propose an *apprenticeship learning* approach to the problem of learning a generalizable robot motor skill that achieves contextually adaptive object hand-overs. The strategy combines learning from demonstration (LfD) and reinforcement learning (RL), such that the former enables the robot to learn the action from an expert's example, and the latter facilitates subsequent experiential learning of contextualized policies, with which different derivatives of the demonstrated action are executed according to what is most fitting for the current context. This method mimics the way we acquire motor skills: grasping the 'ideal' way to perform an action, then learning to adjust it to different contexts as required.

We implement the object hand-over action on the Toyota HSR robot, which must learn to present a grasped object to a person most suitably. In this use case, we consider factors such as the posture of the receiver: whether they are standing, sitting, or lying down, as one of the contexts of the task, since various studies have distinctly identified it as a variable of prime importance to consider for hand-overs, according to users ([114][58][99]). Other factors could include object fragility or potential spillage (for drinks), the presence or absence of an intermediate surface, and so on. The task is characterized by the movement of the object from the initial grasping position to a chosen hand-over position along some trajectory, and the particular execution should adhere to the aforementioned context parameters.

Figure 1.2 illustrates the basic stages of our proposed approach to learning a context-aware hand-over skill.

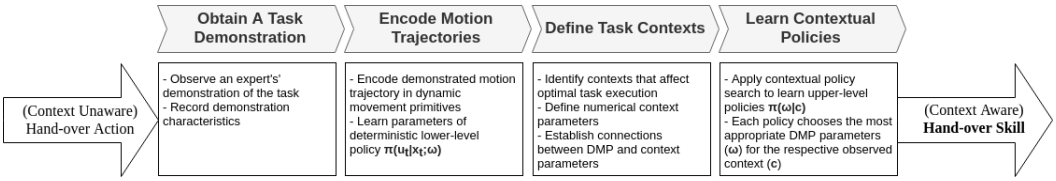


Figure 1.2: The main stages of the proposed apprenticeship learning procedure to learning a context-aware hand-over skill.

The first step of the approach is to acquire an expert's demonstration, which must be captured in some manner and encoded in an appropriate representation, as the robot's initial hand-over policy. Using an existing approach ([76]), we utilize motion capture to demonstrate a trajectory shape to the robot, and encode the motion in dynamic movement primitives (DMPs). DMPs enable recording and reproducing goal-directed motions such as that of the hand-over, as well as altering the goal position, trajectory shape, and execution speed to generate unique execution variants. Among their various favourable properties are spatial and temporal invariance, their compact representation of trajectory shapes, and their extensibility to capabilities such as real-time obstacle avoidance.

Learning from demonstration is a natural learning paradigm often utilized by humans in which knowledge of experts can be exploited to greatly speed up learning of complex tasks, in particular. It

has been successfully applied to robots in various motion-related tasks, serving to similarly make efficient use of expert knowledge, as opposed to manually programming desired behaviour, which is often tedious and produces motions that do not seem as natural. When applied to hand-overs, for example, LfD has been shown to produce more 'natural, usable, and appropriate' executions than those produced by a conventional planner ([20]).

The problem then involves learning to select the DMP parameters that produce the most appropriate reproduction of the hand-over trajectory, according to the current context. In essence, we provide the robot with the necessary contextual knowledge to learn specialized variants of the demonstrated trajectory, by extending the initial, static hand-over policy to a contextually adaptive one. To that end, we propose a contextual policy search approach.

We present a model-based version of the Contextual Relative Entropy Policy Search (C-REPS) [89] algorithm which facilitates learning contextualized policies initialized from a demonstration.

The application of RL is based on the premise that robots which are equipped to explore and learn from experience possess a comparatively more literal form of 'autonomy'. Additionally, learning by reinforcement after observing a demonstration is an instinctive skill acquisition strategy: learning both from a teacher and from personal experience, hence the term apprenticeship learning. As opposed to value-based RL, a policy search (PS) approach is adopted for its proven adequacy for robotics applications and high-dimensional search spaces, in which value function computations become a significant burden ([35][70][101]). Moreover, PS enables seamless integration of initial expert knowledge, in the form of demonstrations, requires fewer parameters, and better deals with continuous action spaces ([88][59]).

Our model-based implementation of C-REPS, inspired by the GPREPS algorithm ([66]), employs contextual policy search, which attempts to learn a policy that is optimal conditioned on the values of a context parameter, through a hierarchical structure of policies. A stationary lower-level policy, in the form of DMPs, is parameterized by the output of an upper-level policy, which selects the meta-parameters of the former according to the current context. As a result, the lower-level policy is made generalizable to different contexts: we learn hand-over position and trajectory parameters that satisfy the context values which signify observed receiver posture, for example. The algorithm utilizes an information-theoretic policy update strategy, which bounds consequent updates in the search distribution to mitigate unstructured exploration and limit information loss, thus retaining the main characteristics of the demonstrated execution.

Crucially, we implement a model-based version of the algorithm, such that we can learn contextualized policies in simulation, before transferring them to the robot. This strategy is often applied in complex RL problems to boost learning speed and reduce possibly prohibitive system interaction time, thus conserving time and effort, minimizing expenses, and eliminating safety concerns. It is often termed *mental rehearsal* to draw parallels with the way humans mentally rehearse an intended action before taking it. Contrary to initial plans of employing Gaussian Process (GP) models, we alternatively utilize a DMP implementation that enables simulating hand-over trajectories and their expected outcomes (hand-over position, trajectory, etc.), which we use to drive the learning process, in a relatively simpler procedure.

The results of our implementation in simulations are then evaluated through quantitative and

qualitative analyses, both to study the characteristics of C-REPS and to determine how well it produces the intended behaviour. We particularly examine the algorithm's policy representation and update strategy, its properties, and its limitations, in addition to the number of artificial executions (roll-outs) required to achieve good results, the number of parameters to be learned, exploration behaviour, and other relevant metrics. With regards to the algorithm's results, the output of the learned policies, we measure the quality of emergent behaviour based on closeness to the desired, and its handling of the multitude of contexts, under different reward function formulations. These assessments are made based on the algorithm's inherent measure of performance: accumulated rewards.

Finally, we transfer learned hand-over policies to the robot, and evaluate resultant behaviour and its implications on user perceptions. In order to empirically validate the hypothesized superiority of contextually adaptive behaviours learned through apprenticeship learning over an antithetical approach, we conduct a user study. The study involves participants assuming different postures and being handed objects over by the robot, under a context-aware behaviour setting, in which it executes hand-overs guided by learned contextual policies, and a context-unaware behaviour, in which it simply imitates the demonstrated movement.

We conclude that behaviour under a contextual policy that selects hand-over positions based on receiver postural context, as learned in simulations, is strongly preferred by users over the traditional alternative, especially in factors such as naturalness, perceived suitability to context, and likeness to human executions. We demonstrate how this result accomplished solely through a demonstration, a simple form of contextual knowledge, and guidance through a scalar reward signal by C-REPS, which achieves these results despite being currently inadequate for some aspects of contextual trajectory learning, as we discuss in the report.

2

State of the Art

In this chapter, we present a systematic review of the literature concerning the multi-faceted problem addressed in this report: generalization of a robot object hand-over skill to different contexts. This review of the state of the art covers pertinent research of influential publications as well as more recent works whose findings, extensions, and innovations merit a methodical analysis. The aim is to provide a reasonably broad perspective on the pool of possible approaches to this end, and judiciously highlight their respective strengths and limitations.

We start in section 2.1 by introducing the concept of robot motor skills, how it may be realized, and the diverse methods with which they can be learned and made adaptive to concepts. This is further divided into sub-sections: we survey the use of explicit contextual knowledge and context-based architectures in robots in subsection 2.1.1, pure control approaches that achieve contextual adaptivity through more implicit means in subsection 2.1.2, leveraging human demonstrations to facilitate imitating skills in subsection 2.1.3, and the promising concept of *apprenticeship learning*, which augments imitations with reinforcement learning, in section 2.1.4. Finally, section 2.2 reviews a range of studies and notable work relating to our representative use case: human-robot object hand-overs.

2.1 Robot Motor Skill Learning and Generalization

As service robots, particularly humanoid ones, rapidly approach integration into households and workplaces, their possession of competences and cognitive qualities similar to ours becomes a natural requisite. One such aspect is that of *motor skills*: intricate movements performed with the intent of achieving some goal or accomplishing some task, characterized by acquisition and refinement through learning processes, and being generalizable to reasonably novel situations. They can range from gross motor skills such as walking and kicking a ball, to finer ones such as handwriting and playing a guitar. Although recently applied with demonstrable success in robots for tasks such as learning to walk (hexapod) [24], playing table-tennis [66], and pouring drinks [119][83], equipping robots with the entailed capabilities required to achieve such complex behavior remains a significant challenge.

In a broad sense, the difficulty of designing adaptive robot skills can be attributed to three constituent parts of the problem: action representation, efficient learning, and generalization to multiple operating conditions. The issue of learning desirable behavior is preceded by the question of how that physical behavior is to be embodied on the robot, with representations ranging from conventional

parameterized controllers to movement/motor primitives [52] (see section 2.1.3). In addition to enabling the reproduction of desired movement trajectories or patterns, skill representations must be compliant to behavioral adjustments. This facilitates applying some form of learning procedure, in direct analogy to human experience acquisition, such that an autonomous robot is capable of iterative improvement over time and, possibly, independent skill development. A significant challenge in its own right, implementing adequate machine learning techniques to realize these desiderata is further compounded by the fact that skills must be robust to changes in the context of a task. This concept of *contextual adaptivity* suggests that motor skills and actions should be applicable to a range of different contexts or situations that the robot could conceivably encounter in its attempts to achieve some goal, in a manner that does not necessitate drastic measures such as re-learning the behavior.

Apart from the understandable appeal of adaptive robot skills to the human intuition, the pursuit of this non-trivial approach to 'intelligent' behavior is justified by more practical considerations. The nature of truly autonomous service robots and the environments in which they are envisioned to operate, by virtue of their unpredictability and high margins of uncertainty, make robustness particularly difficult. In human-oriented applications, a special form of task variability which dictates the optimal way to perform an action is born out of the multiplicity of situational contexts, due to which non-adaptive behaviors may be unsatisfactory. At best, they may lead to user inconvenience and discontent; at worst, they may cause execution failures. For instance, various studies on user preferences for object hand-overs have supported the hypothesis that factors like posture, approach direction, individual capability, etc., should be taken into account by a robot [58][16][15]. It follows that programming a robot to execute the same motion or action: equipping it with a *static* skill, for all cases it may face, is undesirable. On the other hand, learning separate modes of task execution for each context is inarguably inefficient. The compromise, then, is to have a single adaptive representation of the motion(s) required to fulfill a task, in the form of a motor skill that is tolerant to expected situational variations.

Having established this connection between the generalizability of motor skills and the concept of contextual adaptivity, we delve into a rich body of literature which reveals multiple potential approaches. Section 2.1.1 reviews the direct application of *context-awareness* in robot system design, and the explicit incorporation of contextual knowledge (CK) and reasoning processes that guide behavior selection and adjustment, as task contexts vary. In section 2.1.2, an orthogonal approach to skill adaptivity is considered, which endeavors to apply conventional optimal control methods to implicitly build tolerance to contextual variance into the system. In favor of a more human-guided skill acquisition process, the works presented in section 2.1.3 advocate the use of *Learning from Demonstration (LfD)* to directly teach robots the skills in the desired manner of execution, by capturing demonstrated movements in robust motor primitive representations, with natural extensions to enable contextual differentiation between demonstrated movements. Finally, section 2.1.4 further builds upon this concept, introducing reinforcement learning techniques as a paradigm both for independent learning of motor primitives, as well as optimizing and generalizing those learned using LfD to new situations. These techniques are of particular interest, owing to their similarity to the way motor skills are realized in humans: a combination of muscular activity (motor primitives) and cognitive processes (such as reinforcement learning).

2.1.1 Context in Robotics

Various researchers and research groups have highlighted the importance of equipping robots with representations of situational contexts, as humans would understand them, particularly when their operations involve interactions and collaborations with humans, especially if they are to stand a chance of being regarded as 'intelligent subordinates' ([120]). Nevertheless, when it comes to utilizing this concept in robot behavior, like many other human-centered concepts, the idea of context suffers from the curse of ambiguity: we can easily perceive its impact on the way we conduct our behavior in daily interactions, and would readily recognize its significance, but most people would struggle to articulate a coherent definition of context. Bloisi et. al., who investigate the use of context in robotics, provide in [19] two definitions of different granularity: context can be thought of as "the information that surrounds a situation of interest in the world", and "an identifiable configuration of features which are meaningful to characterize the world state, and useful to influence the decision process of a robotic system" (or human).

One of the earliest principled works concerning design for context-sensitivity introduced an approach named *context-mediated behavior (CMB)*, based on contextual-schemas (c-schemas): knowledge structures containing descriptive knowledge about a context, and prescriptive knowledge about the implied optimal behavior. In [108], Turner et. al. lay the groundwork for this explicit treatment of context in artificial intelligence, which would later influence a significant portion of the reviewed literature. In an AI-oriented perspective, they define context as "any identifiable configuration of environmental, mission-related, and agent-related features that has predictive power for behavior". Motivated by the ideas that context-free behavior can not be generally 'appropriate', and artificial agents that do not adhere to contexts are useless, they propose recording 'features' of a situation in c-schema that encode a coherent view of the context of a task, and specifying a mode of execution that best suits it. The generalization, then, lies in the agent's ability to match perceived situations to c-schemas stored in memory, possibly merging information from multiple c-schemas to formulate novel contexts, and subsequently using this information to augment other reasoning processes. The possible uses of this contextual knowledge include better interpretation of sensory data, auto-modulated behaviors, focusing attention on relevant goals, and selection of actions appropriate to the context. A demonstrated application is to an intelligent mission controller for AUV oceanographic missions, Orca: a schema-based, adaptive reasoner composed of multiple modules, one of which is a CMB context manager that optimizes AUV task execution with respect to context. Although the framework reduces planning burden through context-awareness, can reasonably generalize to unseen contexts, and could facilitate autonomous adjustments of contextual knowledge with experience, its applications may be limited to fairly similar systems. It relies on explicit enumeration of context features in a Common Lisp Object System (CLOS) framework, and does not seem compatible with motor skill refinement on a robot, for example.

A decade later, context-based architectures (CBAs) for robotic systems, founded on the formulations of context put forward by Turner et. al., were implemented as described by Calisi et. al. in [22]. In a similar approach, the authors aimed to exploit the use of CK: available information to the system, which characterizes the task-related context of the current situation in which the robot operates. Their objective

was to adjust the behavior or functionalities of a mobile robot dynamically to fit the perceived situation, hence improving its adaptivity to changing conditions. Referring to past approaches that used CK in individual tasks including navigation and exploration, localization and mapping, and perception, they proposed an architecture for system-wide contextual-sensitivity, stating that CK is not fully exploited when it is built into separate system modules. This system-agnostic CBA is designed as a feedback loop added to some robotic system, inserting modules that convert sensory data into some internal representation of context, reason about it, and convert it to direct commands that subsequently influence a robot's control policies. Crucially, the reasoning elements are designed such that they utilize CK in driving and adapting different tasks that the robot must accomplish, making its compound behavior adherent to the context. The benefits of the CBA, and its easy implementation on existing platforms, is demonstrated on a search-and-rescue robot, whose SLAM modules are augmented with the design, and its performance is shown to improve in efficiency for an exploration and search task, as well as a navigation and mapping experiment. The applicability of the architecture to different platforms and tasks, however, remains doubtful since it was only evaluated on the mobile robot in the scenario chosen by the authors. This is exacerbated by the requirement of having a robust representation of context that fits the reasoning module, which may be fairly complex to ground for certain applications.

In [38], Gomez-Romero et. al. explore the use of CK in information fusion (IF) applications such as video surveillance, ambient intelligence, and context-aware systems. In their work, the authors apply a 'context layer' in a computer vision framework, define an ontological representation of context information, and an inference engine for abductive reasoning about perceived scenes, with the aim of justifying context exploitation strategies in the aforementioned applications. Despite not focusing on robotics, the authors highlight various challenges that directly correspond to those in our present work, including whether context is to be represented quantitatively, symbolically, ontologically, or otherwise, and understanding that the choice affects the nature of the implemented algorithm or learning process, which is usually forced to cater to it. While empirical results derived from this study reflect the power of ontology-based context representations and reasoning, the evidence is ultimately deficient in proving its suitability to adaptive robot motor skills, a rather incompatible problem.

Context-aware selection of behaviors/actions can also be paired with programming by demonstration, in a perception-based perspective on context differentiation, as illustrated by Narayanan et. al. in [80]. This work involves identifying different contexts of a scenario, through matching observed perception feature histories, and choosing a particular, demonstrated behavior, acquired through partial tele-operation, that best suits the identified context. Here, this is applied to visual-based navigation of indoor environments with varying obstacle formations, which the authors claim is bound to fail if a monolithic model of the relations between individual perceptions and actions is used, due to the variability. An idea of contexts is then introduced as sets of visual perceptions along with the curvature of the recent path of the robot. This visual robot behavior learning achieves a form of context-awareness that increases its robustness to a range of controlled situations, but remains dependent on perceptual data, a form of feedback that may not apply to applications that involve different sensor modalities.

The same formalism used in the seminal work on context-mediated behavior (CMB) ([108]) has

since been extended to a multi-agent system scenario in [109]. The importance of context for teams of agents was motivated by factors such as the complex scenarios they may operate in, their heterogeneity, dynamic re-formulation of team composition, and compensating for collaborator shortcomings, all of which necessitate context-dependent abilities of adjustment. Additionally, identifying global context can enable individual agents to better interpret sensory information and choose actions that fit the context best, with respect to actions of the party. Interestingly, the aforementioned concept of *c-schemas*, including all its desirable properties, is shown to be directly applicable to this more complicated scenario, notwithstanding expected issues of multi-agent communication: distributing context assessments, merging CK and using it effectively, and so on. A major portion of the authors' work involves choosing an appropriate language and shared ontology for multiple agents to communicate with and form a global understanding of context.

An intriguing approach to the use of CK applies it to accomplish context-aware grasp planning in shared autonomy, as presented by Witzig et. al. in [118]. The main motivation is similar to our own, but in the context of grasping: a robot's manipulator is capable of a variety of potential grasps of an object in a described scenario, but some may be better applicable to some contexts than others. The authors introduce a probability distribution on a set of enumerated grasp type hypotheses , which consider reachability constraints and collisions, among which the best is picked according to the context. In this case, the CK is a combination of high-level information provided by a user through a novel dialog-based UI (task type, container fill level, etc.), and auto-generated information derived from online perception (object size, side graspability, etc.). The concept of shared autonomy comes into play in the learning process: the system enumerates possible grasp poses and displays them to a user, who selects positive examples for the given context, then updates its probability model to reflect the user's suggestions. In addition, if the robot fails to find an appropriate grasp during normal execution, the user is consulted through the UI to provide a suggestion, with which the model is refined. This helps in tackling scarcity in training data, and reducing failures of error-prone auto-generalization, but seems to suggest limitation to the supervised learning paradigm.

A recent publication by the research group who published [22] provides a novel classification of existing context-representation methods in robotics, and a context-aware framework for information fusion (IF) ([19]). The authors re-iterate the importance of contextual adaptivity in robotics in order to accommodate multiplicity in operational conditions, referring to its proven usefulness for cognition-based systems, by improving performance and scope of applicability through context representations and subsequent reasoning. It is interesting to note that, in direct contradiction to a sentiment expressed in their previous work ([22]), they advocate using context in individual components of a system, where needed, as opposed to the usual holistic approach, which can be taken as an indication of the difficulty in designing for contextual adaptivity. When it comes to robots, CK is taken to be the sum of environmental knowledge (lighting, terrain properties, etc.), task-related knowledge (constrains like time, priorities, locations, etc.), and self-knowledge (e.g. of internal status). Applications to which each of these aspects of context-sensitivity is demonstrated are described, including surface maps for terrain conditions (E), robotic walkers for parkinson's patients (T), and robot behavior specialization through smooth transitions between behaviors, as dictated by CK of system status (S). Valid and implemented context representations

are grouped into embedded, logic-based from (rule-based ontologies to first-order logic), and probabilistic (such as Bayesian Networks) approaches. The major contribution of this work lies in its extensive survey of these perspectives, the choice of which may constitute the biggest obstacle in incorporating CK into robot task execution.

2.1.2 Control-based Approaches

A radically different, but fairly plausible, solution to the problem of adaptive robot motor skills does not rely on explicit representations of context, but designs adaptivity to changing contexts implicitly, using optimal controllers which satisfy specified criteria. Overlooking the virtual elimination of the learning aspect with these techniques, one may rightfully argue that the reviewed works accomplish some form of contextual adaptivity, even if it does not entail context-awareness, which justifies consideration of this approach as a valid alternative. Here, we sample and review some demonstrations of this approach but with a focus on those relating to our use case, both to ensure reasonable relevance and to constrain the otherwise vast reserve of implementations in this vein of research.

The work by Sisbot et. al. in [99] presents a robot manipulation planner that takes human presence and activity explicitly into account, producing motions following reasoning about a person's safety, field of view, accessibility, postures, and preferences. The planner is applied in the context of a robot-human object hand-over task, and focuses on enabling the robot to compute and execute the best human-oriented, or 'socially acceptable', trajectories. The authors attempt to study what can be done at the motion planning and control level, in order to facilitate awareness of the human, and subsequent behavioral adaptation. To this end, the planner computes the best object transfer position, as a cell in a 3D grid overlaid around a person, which maximizes three crafted cost functions representing safety, visibility, and human arm comfort, while taking the path of the object and required robot motions into account. Experimental results of user case studies, in which comparisons are drawn with a baseline planner, illustrate the improvements in resultant robot motion trajectories, which are validated using subjective (user feedback) and objective measures. The incorporation of the person in this planning process is significant, due to the importance of prioritizing safety in shared-manipulation scenarios [82], and the proven preference of users for motion-level robot adaptation, particularly for human-robot joint performance fluency [68]. A consequence of these optimized cost functions is minimum-jerk, more human-like hand-over trajectories which makes them superior to more rough, unpredictable ones [90], possibly produced by baseline planners, and whose smoothness properties are known to be decidedly preferred [84]. Nevertheless, this manual optimization of cost functions, after which controller behaviour is virtually static, exposes a latent issue: the criteria defining good manipulation trajectories varies across users, tasks, and environments [53], and is fairly difficult to anticipate during controller design [54]. Another conceivable drawback is the exclusion of biological differences among object receivers, who are treated as 'average' individuals with similar physical capabilities. However, studies show that generic behaviour in tasks such as object hand-over are made inapplicable [47], due to situation-dependent differences such as the perceived ability of the other party [102], and ergonomic preferences and/or motion restrictions due to age, injury, disability, and so on [15]. These factors make it difficult to argue that the resultant behavior has the ability to conform to different

contexts, an outcome we ideally aim to achieve.

A more novel approach similarly attempts to generate 'optimal' hand-over positions, whose quality is quantified by the amount of discomfort of, or forces having to be exerted by, the person being handed an object, utilizing constructed bio-mechanical models, as presented by Suay et. al. in [102]. These individual-specific custom models guide the search for a hand-over position which considers the person's range of motion and static strength limits, addressing one of the deficits of [99]. The authors' algorithm generates this obstacle-free point on-line, starting with taking in the person's gender, height, and weight as inputs, and constructing a bio-mechanical model using OpenRAVE, a robot motion planning simulator. They highlight that sophisticated human models that include muscle dynamics exist, but are rather superfluous for the purposes of their work, since such a simple model can enable sufficient hand-over position analysis. Using OpenRAVE tools, they derive joint forces and moments in a simulated reaching out motion for a generated bio-mechanical model, using an inverse kinematics solver. Subsequently, candidate hand-over positions are sampled, filtered by performing a k-NN search to find the positions most resembling what was observed in human demonstrations, and optimized to account for perceived obstacles, etc. In a comparative analysis, the authors conclude that the peak muscle activity recorded for a person while reaching out to a generated position was interestingly less, on average, than that of an arbitrary position chosen by the person themselves. Catering to different people's physical capabilities and constraints is an important consideration to account for in robotics, particularly for human-robot interaction, and makes this form of skill adaptivity a vital one for service robots. It is also worth noting that the low computational overhead of the authors' algorithm, considering the low number of parameters they require as input, makes it particularly useful in such applications for which on-line computations are necessary. However, these very inputs raise the question of autonomous adaptivity, since they would have to be manually input by a person, and the robot would lack the ability to truly adapt autonomously during operation. In addition, reliance on the bio-mechanical models that can be made using OpenRAVE may introduce restrictions on representable contexts and limit practical extensions to more complex problems.

A novel human-inspired motion controller for bi-directional human-robot object hand-overs, introduced by Medina et. al. in [74], is the result of an attempt to produce a hand-over dynamics model, with which a controller could produce seamless motions that are independent of the particular phase of the hand-over, an undesirable limitation present in conventional controllers. The authors contend that it is highly desirable for hand-overs, as is the case with any human-robot collaborative task, to be as fluid and natural as inter-human hand-overs, for arbitrary objects and situations; an instance of an adaptive motor skill. They point to the traditional hand-over task formulation, which models it as a set of successive, discrete phases (approach, passing, and retraction phases), as being unsatisfactory due to the resultant rigidity in a robot's executed motions. A human-human hand-over study was conducted to gain actionable insights on the dynamics of natural hand-overs, which the authors then used to guide their design to produce similar levels of motion fluidity. Leveraging this, and an extensive mathematical formulation of object hand-over dynamics, they produce a controller that is able to reliably estimate hand-over poses, continuously monitor the load share that governs efficient grip force, and being robust to outside interference, mid-execution. With experimental trials illustrating the controlled robot's ability to

appropriately grasp a plastic water bottle, track the receiving person's hand, and apply the necessary grip forces at the adequate time to release and re-grasp it, the authors demonstrate an overall enhancement in task fluency. A noteworthy flaw found in the author's presentation of their solution is a claim that the controller produces hand-overs that are adaptable to different object types and shapes, which is not actively validated or subsequently elaborated on in the paper. Among the notable accomplishments of this work are continuous, minimum-jerk hand-over motions which are better accepted by users [47], impressively efficient grasping and releasing of the object, which mitigates the positioning burden placed on the person interacting on the robot [49], and reasonably fluent, synchronized execution, a persistent challenge to natural human-robot hand-overs [65]. The paper's insightful human hand-over study is a valuable resource, as attested by related publications that advocate the investigation of underlying principles in order to implement more fluent robot executions of the task and provide a credible formalism of hand-overs in general ([49][65]). On the other hand, the method suffers from a lack of contextual adaptivity, since issues like the different postures and positions of the person involved, for which the handover characteristics may have to be altered (during run-time), as proven in user studies like [58] and elucidated in discussions on context in robotics, like [19], are not considered. A familiar issue is indistinction between user capabilities: the force responses and passing phase timings, which are the main parameters with which the design is evaluated, may differ between people of different strengths, health conditions, age, etc. As previously mentioned, behavioral adaptation to users is vital for human-robot interaction [95], especially for assistive robots targeted towards elderly and disabled people, for which the same motions would not be appropriate [37].

Another control-based approach to task adaptivity in a robot manipulation scenario is shown by Kappler et. al. in [56], where real-time perception is combined with reactive motion generation to compensate for uncertainties and unmodeled dynamics when planning motions to pick up objects on a table. In order to investigate the improvements brought about by this integration, the authors compare a trio of possible system architectures in a manipulation planning scenario: sense-plan-act, locally reactive control, and reactive planning. The first involves initially perceiving the environment, planning an optimal collision-free path, then strictly following it with a stiff controller, while the second relies solely on perceptive feedback for local reactivity (as in visual servoing): the local geometry around the manipulator is used to continuously compute the next optimal control command. A hybrid approach, reactive planning, which combines perceptual feedback and also re-planning in the event of significant changes, is suggested and compared against the other two in different, static and dynamic, on-table object pick-and-place tasks. Experiments expose the failure of the sense-plan-act paradigm in dynamic environments (introducing an unmodeled obstacle during task execution), and the susceptibility of short-sighted locally reactive control to local optima, when the task requires longer planning horizons. Reactive planning, on the other hand, provides a better trade-off, such that dynamic environments are handled reasonably well without incurring too much re-planning costs. Although possibly somewhat limited in scope of applicability, this approach demonstrates the virtue of perceptual feedback when aiming for adaptive robot behaviors, and their 'robustification' through more rudimentary means.

As can be concluded from the summarized deficiencies in each of the reviewed works, solely

employing controllers, however dynamic, and designed cost functions is restrictive when applied to the problem being addressed in this project. Since it is somewhat impractical to directly define and control multiple context variables, for example, they may not be well-suited for the desired robot skill generalization. These approaches are usually limited by the fact that controllers must be redesigned or their parameters re-estimated for marginal modifications of a robot's skill, or additional task constraints [96], a problem addressed in part by some of the methods reviewed in the next section. These solutions additionally better deal with the fact that human-like motions, which we tend to prefer, are difficult to guarantee when the robot's behaviour is dictated by planners and conventional controllers. A better improvement still is the introduction of reinforcement learning, seen as a generalization synonymous to *adaptive* optimal control, that does not have the same assumptions of perfect knowledge in system descriptions, and operates directly on measured data and rewards from interaction [59], possibly yielding more adaptive motor skills than pure optimal control.

2.1.3 Learning from Demonstration (LfD)

Learning from demonstration (LfD), also called *programming by demonstration (PbD)*, or (the biologically inspired) *imitation learning*, is a skill acquisition procedure and supervised learning framework with which a robot can learn complex control policies or behaviors, usually movement trajectories, from expert demonstrations. The technique was introduced to mitigate the tedium of manual programming, and reduce developmental and maintenance costs, especially for complicated and intricate tasks. LfD effectively allows desired reproduction of underlying movements that are possibly hard to explicitly model, as in motor skills. LfD offers a form of bootstrapping when learning and improving on complex behaviors: it acts as an initialization for imitation-driven learning, or as prior knowledge introduced into a control policy, to effectively reduce search spaces and increase learning speed. With deep inspirations from biological learning processes in humans, the use of LfD in robotics poses non-trivial questions of what, how, when, and who to imitate: the former two can be addressed from the perspectives of learning and encoding skills, respectively, while the latter two remain largely unsolved. A comprehensive overview of LfD by Billard et. al. can be found in [17]. For a more recent survey of LfD in robotics by Argall et. al. encompassing the terminology, common issues pertaining to demonstration techniques and media, policy derivation from demonstrations, and a categorization of approaches, the reader is directed to [9].

LfD, by definition, presents a very suitable method for encoding human-like motor skills in robots, and offering a starting point for autonomous improvement from there on. Among its benefits is the reduction in search spaces for skill learning, directly affecting convergence and quality factors: demonstrations enable starting from a promising subspace and eliminating conversely undesirable, or even futile, regions. In policy-based reinforcement learning problems, demonstrations provide a form of 'pre-structured policy' that can increase problem tractability, next to neural networks, controllers, and linear models [104]. As previously mentioned, LfD may also minimize requirements for tedious programming, possibly increasing accessibility and user-friendliness of robots, to lay people, in a more natural instruction paradigm. Human demonstrations have previously been encoded on robots using kinesthetic teach-in, remote control of motor commands, and motion-capture set-ups (magnetic or optical

tracking; similar to what is used in the present work). The skills obtained from demonstrations can be encoded in various ways: symbolically in pre-defined sets of controllers, on the trajectory-level using statistical models like GMMs and HMMs, or in well-founded motor primitive formulations (see section 3.2 for a more detailed discussion). Learning demonstrated skills is, however, not devoid of complications such as correspondence problems, robustness to perturbations, and potential for generalization, but the literature reveals sustainable solutions to address each. Ultimately, LfD proves to be an invaluable technique in the pursuit of learn-able, generalizable, robot motor skills, which warrants a review of the concerned literature.

In [21], Calinon et. al. already demonstrate a viable application of LfD in a framework that allows a robot to learn the latent details of a demonstrated manipulation task, reproduce it, and generalize it to different operating conditions. From multiple demonstrations by a human, as they guide a humanoid robot's manipulators through kinesthetic teach-in, the relevant features of the task at hand are extracted by projecting joint motion data onto its principal components, using PCA. A probabilistic representation, in this latent space, is encoded in Gaussian and Bernoulli Mixture Models, which are used to find the relative importance of each motion variable (joint speeds, etc.) and dependencies across them, yielding a time-dependent similarity measure with which the motions reproduced by the robot can subsequently be evaluated. The algorithm then generates trajectories using Gaussian Mixture Regression (GMR), which aim to optimize this measure for a certain 'context' (i.e. different operating conditions of the task), given the robot's kinematic constraints and the position of objects to be manipulated. Here, GMR reconstructs trajectories by estimating the conditional expectation and covariance of the spatial values of each motion variable through regression, given consecutive temporal values as query points. Experiments with the small Fujitsu HOAP-2 robot performing tasks like moving a chess piece, lifting a bucket, and picking up and moving an object, show the extraction of variable correlations and the reproduction of the motions for different conditions, such as initial object position and using the robot's other, un-trained arm. This method for motion learning in a lower-dimensional subspace would prove to be a popular approach to LfD, relying on statistical models for skill encoding, as in this paper, and eventually stochastic processes, as in other reviewed publications.

An exemplary instance of using LfD to learn and generalize motor skills for robot manipulation tasks can be found in [83], in which Pastor et. al. present one of the earlier applications of dynamic movement primitives (DMPs) to such scenarios. DMPs (which are more thoroughly discussed in section 3.2) enable learning and generation of discrete and rhythmic movement trajectories, and are encapsulated in second-order differential equations. This formulation enables efficient manipulation of goal-directed motions, including reshaping trajectories, spatial and temporal scaling, and goal (end-point) generalization. Pastor et. al. demonstrate the use of DMPs for generalization of demonstrated trajectories by altering goals, with robustness to perturbations, presenting a revised version of the DMP equations that tackles some numerical issues of the original formulation. Additionally, they suggest and implement sequencing of individually learned DMPs that encode 'primitive' actions, to enable synthesis of compound, complex tasks (grasp-place-release), subject to considerations such as starting a subsequent motion before the first ends, such that jerky transitions are avoided. A drink pouring task is learned and made adaptive to

different object positions as follows. An anthropomorphic Sarcos Slave arm is taught motions for grasping, pouring, retracting, and releasing through demonstrations by kinesthetic teach-in, with the resultant DMPs being stored in a library. For a drink serving task, these are sequenced manually and executed to perform the complex skill, for different bottle and cup starting positions on a table. A second task involving moving a cup between two arbitrary positions on the table, while an obstacle interferes with the planned trajectory, is performed to demonstrate online adaptation through obstacle avoidance, which is enabled by simply adding a coupling term to the differential equations ([45]). This work exhibits the power of DMPs for encoding desired movements, reproducing them robustly, generalizing to different contexts (at least when they manifest as goal attractor positions), and sequential compound movement generation, making them highly suitable for representing robot motor skills, as we consider them in the present work.

Similarly of the opinion that robot motion skills acquired through imitation must be adaptive to different 'contexts', Ude et. al. present in [110] a novel method to make skills encoded in DMPs, as in [83], generalizable to unseen situations, using only provided demonstration data, as in [21]. However, in contrast to such works that project this training data into a lower-dimensional latent space, possibly causing over-smoothing and loss of relevant information, the authors take an uncommon approach: computing new DMP (control policy) parameters online, directly from data samples. In essence, they tackle the problem of generating DMPs, expressed in terms of shape parameters w, temporal scaling parameter τ, and goal (attractor) parameter g, for new 'situations' represented by task parameters q, given demonstrated trajectories corresponding to some observed values of q_k, as training data (note that these task parameters are synonymous to context variables; the term is used here to adhere to the authors' terminology). To learn a mapping, $q \rightarrow [w, \tau, g]$, the online learning procedure uses Locally Weighted Regression (LWR) to estimate w, and the less computationally tame Gaussian Process Regression (GPR) (see section 3.3, for an overview on GPs and GPR) to estimate g and τ. This unknown functional relationship between task parameters q, which are used as query points, and the optimal DMP parameters is thus learned, such that motions not initially part of the example database can be generated to address novel contexts of the particular task. The validity of this approach is proven through experiments, among others, where humanoid robots, HOAP-3 and CB-i, perform reaching tasks with and without integration of visual feedback, respectively, with task parameters representing desired Cartesian positions. As far as one can judge, notwithstanding the impressive skill generalization ability of the authors' method, the assumption of adequate DMP parameters being a smooth function of task parameters q, and the reliance on demonstrations being sufficient to approximate this mapping, using a Gaussian process, suggest conceivable difficulties in practice.

An interesting extension of this work was shortly proposed by Forte et. al. in [36], where the authors identified the prohibitive computational cost associated with the algorithm running in a real-time feedback loop. The alternative presented to tackle this involves substituting conventional DMPs with dynamic systems that encode a whole class of movements, in the form of a more general non-linear system of differential equations. Similarly, the mapping between task parameters q and those of this complex function is realized using GPs, with GPR enabling real-time computation of this general equivalent

of DMPs in a manner more conducive to real-time sensory feedback loops. In addition, the previous memory-based approach, which would store demonstrated DMPs in a database, is improved upon to allow on-line switching between motion primitives, given sensory feedback, through the use of this formulation. Despite the notable enhancement, general, aforementioned limitations may still apply.

LfD need not be limited to relying solely on position data, and has been effectively applied with force sensing to learn adequate compliance behaviors through adjustable impedance, facilitating physical human-robot collaborative tasks. In Rozo et. al.'s [96], the importance of force sensory information and variable impedance in these scenarios motivates utilizing LfD to learn complex collaborative skills that incorporate time-varying compliance levels, and subsequently modulating behavior based on user actions and task parameters, enabling collaborative object transportation and table assembly tasks, for example. Here, demonstrations are encoded probabilistically in task-parameterized GMMs (TP-GMM), where task parameters: variables in the form of reference frames attached to initial, goal, human, object, and obstacle positions and orientations, that influence robot behavior, transform the model parameters (mixing coefficients, means, and covariances) to induce appropriate trajectories. In addition, each Gaussian component is coupled with a local stiffness matrix, capturing the local dynamics of the action. Conducted experiments demonstrate that a robot can successfully capture the dynamics of a task, including 'stiffness profiles', and reproduce it while handling varying position and force constraints, imposed both by the environment and reasonable human-induced variations in the task. The presented framework lends itself well to human-oriented applications, where safety is a significant concern, and demonstrates a more advanced manifestation of skill generalization, owing to the novel integration of force modalities. While the defined task parameters work well in the tested scenarios, restricting them to mere reference frames of task-relevant objects around the robot is potentially limiting in expressing and differentiating between contexts, from a more general perspective.

Choi et. al. ([26]) enhance the robustness and user-friendliness of the typical LfD process, by accommodating potentially unreliable demonstrations such that the assumption that they always come from skilled experts is relaxed. Also using GPs to encode demonstrated movements, they introduce leverage parameters that weight the reliability or importance of training samples, which are used to score demonstrations in the range $(-1, 1)$. The resultant leveraged Gaussian processes encode both positive and negative examples in a unified regression framework, with the leveraged GPR being used to approximate the optimal behavior using a "sparse-constrained, leveraged optimization" procedure. This method is shown to outperform baseline LfD methods (namely, LfD-GP and LfD-kNN), in a planar navigation policy learning problem. The novelty of this work lies in addressing latent shortcomings of the LfD approach, when applied to instruct robots in a sustainable way, including the dependence on possibly unreliable 'expert' demonstrators, or lack thereof, and the foreseeable decline in the quality of behavior as different demonstrators contribute to the robot's skill acquisition process, to name a few.

When it comes to tasks involving rigid body motion trajectories and contextual differences in terms of object positions and execution speeds, the invariant trajectory representation of a demonstrated skill, introduced in by Vochten et. al. in [113], provides interesting advantages over others. This representation extracts rudimentary kinematic motion features, aiming to remove context-specfic information not

necessarily inherent to the demonstrated motion, such that it can be generalized to a wider range of contexts without the need for too many demonstrations. Frenet-Serret *invariants*: coordinate-free descriptions of trajectories, are used to "eliminate the dependency of the trajectory coordinates on the choice of a reference frame in which the coordinates are measured, and the initial position and orientation (pose) from which (an) object starts moving...", as put by the authors. They extend these further to avoid dependence on execution style: velocity profile, duration, and scale of motion, by introducing a scalar progress parameter, referred to as the 'degree of advancement'. Dynamical system equations are formulated to allow both reconstructing demonstrations and generating motion for novel contexts using these dimensionless, geometric invariants, with the latter made possible by using the equations in a constrained optimization problem, involving constraints relating to the contextual information. In the interests of brevity, mathematical specifics are omitted here. While other representations incorporate some invariance measure, such as time-invariance in DMPs, the resultant coordinate-free trajectory representation is thought to maximize contextual independence, eliminating all but essential motion information. Learning and generalization capabilities are evaluated in a simulated pouring task, in which different target positions, various initial object orientations, and even mid-execution change in target positions are all reliably adapted to. The authors present empirical evidence suggesting the superiority of this approach to DMPs, since it seems to better preserve trajectory shapes, as contexts vary. Another major merit is the ability to reasonably generalize given a single demonstration, since the invariants theoretically provide the basic, intrinsic features of a trajectory. It is worth noting, however, that the focus of this representation on rigid body motion problems means representations of context may be limited to those conceptually adjacent to, if not synonymous with, object and target positions.

Shared control, or assisted teleoperation, frameworks have been augmented with LfD in [5] by Abi-Farraj et. al., with the objective of leveraging demonstration data to learn optimal on-line adjustment of the degree of robot autonomy, in contrast to following human preference. Given multiple demonstrations of a task, a distribution of trajectories is learned, with low variances signifying particular preference on the motion, and higher variance implying more relaxed requirements. This is used to drive the balance between autonomy and tolerance for human intervention (or teleoperation), with areas of low variance leading to stronger resistance to deviation from what is considered the nominal trajectory, according to the distribution, and the others allowing more freedom in manually guiding the robot. Resultant behaviors improve through continuous interactive executions, since the data from each is aggregated such that the trajectory distribution is constantly refined. A linear-Gaussian model is used to model the distribution of trajectory positions at each time-step, conditioned on a context variable that later enables generalizing to new situations. In essence, the confidence of the robot in performing some task, given the context, is embodied by the distribution, which intuitively depends on how varied or, conversely, how exact and meticulous the demonstrations by the person(s) are. The approach was validated in experiments involving a robot manipulator master-and-slave setup, with which an object grasping task was taught and partially performed by a human. The robot was compliant to the person's intervening inputs, as it reproduced the demonstrated movements, but provided informative force feedback cues to signify their deviation from the learned behaviour, facilitating an effective shared control architecture.

Lundell et. al. introduced global parametric dynamic movement primitives (GPDMP) to enhance the generalization capabilities of traditional DMPs, by utilizing reinforcement learning [71]. Although movement primitives and trajectory representations discussed up to this point enable generalizing over task parameters without the need for learning separate primitives for each context, they interpolate across demonstrations much better than they can extrapolate; a limitation specifically targeted by the authors. As in previous approaches, The kinematics of a demonstration are encoded using DMPs. However, the authors then use the PoWER algorithm (see section 2.1.4) to optimize the shape parameters of the DMP such that they adapt well to some different local task parameters. The GPDMP model is constructed from these optimized primitives, employing "a linear basis function model with global non-linear basis functions", effectively creating a global model for mapping a task parameter to policy (DMP) parameters, with better extrapolation to unseen task parameters. The use of reinforcement learning here ensures that the training examples used to create the global model are reliable, as opposed to directly encoded demonstrations, for the given task, and obviates the need for multiple demonstrations to capture the same primitive. The ball-in-a-cup task, a popular motor skill and RL benchmark, demonstrates the generalization performance of GPDMPs to different string lengths: the context variable used by the authors. A KUKA LBR arm is taught using kinesthetic teach-in, and its extrapolation to unseen contexts is shown to be superior to similar approaches. GPDMPs can be improved to handle more than a single task parameter, as the authors state, but demonstrate an application of RL to LfD: a promising approach that will be reviewed in the next section.

As we transition into realizing the potential RL has to improve demonstrated skills, particularly when contextual adaptivity is of concern, we must also consider a reciprocal view: LfD can be an effective tool to overcome characteristic issues of RL algorithms. Such an issue is that of exploring task environments with sparse rewards, in which the difficulty in receiving any reward is exacerbated by long task horizons and prohibitive action frequency and dimensionality, leading to stunted learning. In [79], demonstrations are used by Nair et. al. to guide the exploration process when learning in the context of robotics, such that random exploration is replaced by educated guesses of the promising areas to pursue rewards, facilitating acquisition of good policies early on. By providing an algorithm with this form of 'pre-structured policy', as it was referred to earlier, complex, multi-step tasks that would otherwise be beyond current capabilities can be realized, in addition to vastly accelerating convergence. The authors extend Deep Deterministic Policy Gradients (DDPG): an off-policy model-free RL algorithm that usually uses neural network policies, with demonstrations, and employ Hindsight Experience Replay (HER): a RL method that efficiently utilizes failed roll-outs to guide learning, producing an effective skill acquisition and improvement framework. Experimental analyses are conducted for several robot tasks simulated in MuJoCo, using a 7-DOF Fetch Robotics arm, such as pushing, sliding, and pick-and-place, demonstrating the superiority of initializing actions with expert demonstrations, collected using a HTC Vive interface. The framework's performance is also tested on a more complex, multi-step task involving stacking blocks. The presented combination of RL and LfD addresses the exploration challenge with a satisfying solution, but may suffer from sample efficiency due to the large amount of experience that must be gathered, and is yet to be applied to a physical robot.

To summarize, the LfD framework, including the various trajectory representation and motion primitive formulations, offers an intuitive and well-grounded solution to transferring motor skills to robots. Using GMMs, DMPs, GPs, or the invariant trajectory representation, as examples, has been shown to enable capturing the relevant features of a demonstration adequately, followed by reproduction and reasonable generalization. Strictly speaking, however, imitation learning is primarily concerned with matching demonstrator performance, which theoretically places limits on contextual adaptivity and bounds on performance imposed by demonstration quality. *Consequently, the use of reinforcement learning emerges as a natural extension to this methodology.* As discussed, LfD helps in overcoming problems in RL, even provably transforming intractable problems to feasible ones, by virtue of the aforementioned restrictions on search spaces ([59]). This, along with improving over demonstrated skills, gives rise to various approaches that attempt to harmonize the two techniques, which are included in the next section.

2.1.4 Improvement Through Reinforcement Learning (RL)

The principal reason for augmenting robot skill acquisition with RL lies in an intuitive sentiment: natural imitation should entail more than passive repetition of tasks, ideally incorporating some active exploration in an adaptive learning paradigm [40]. Movement primitives, captured from demonstrations or otherwise, can be refined over time using efficient RL algorithms, which are also applicable for generalization to different contexts, in lieu of previously encountered methods like GMR and GPR. Drawing parallels with skill acquisition processes in humans, the union of LfD and RL has been termed *apprenticeship learning*, underscoring the virtues of learning both from a teacher and independent practice, and of independent rehearsal [59]. In this manner, learned movements and actions better qualify as 'skills', considering the integration of guidance and experience, the latter potentially leading to outperforming the demonstrator's policy, yielding even better results.

Crucially, apprenticeship learning, thanks to RL algorithms that accommodate contextual knowledge in policy learning, can also be well-suited for learning contextually-adaptive behaviour, as will be shown by the publications reviewed in this section. The majority of these incorporate themes relevant to the present study, including robotics, human-oriented tasks, leveraging LfD, and so on. The two subsequent sub-sections target a particularly relevant sub-set of algorithms involving hierarchical reinforcement learning (HRL), and contextual policy search (CPS), respectively.

Robot behavioral adaptation in HRI has long been realized through reinforcement learning for the adjustment of skills or general behavior. For instance, Mitsunaga et. al. present a behaviorally adaptive system that enables interactive robots to leverage social human cues in order to adjust their behavior to match preferences of a person with whom they interact [77]. The authors propose perceiving such signals, particularly discomfort cues, and using them as inputs to a RL reward function, which helps in optimizing behavior to minimize these signals. These include gaze aversion, or irregular repositioning of the body relative to the other party, whose minimization would ensure more acceptable and adequate reciprocal behavior. An experimental setup involved subjects interacting with a robot that produced various behaviors, subject to parameters that can be easily altered and strongly characterized an ongoing interaction (distance, extent to which human gaze is met, waiting time between utterance and an action,

etc.), and adjusted them according to implicit inputs from the user. Using policy gradient reinforcement learning (PGRL), and utilizing its perception, a humanoid robot, RobovieII, was shown to adjust behavior by fitting the parameters to a person's liking, during interaction. This RL-based, human-oriented adaptation, although it marginally addresses contextual adaptivity, provides useful insights. As human subjects attested to, the resultant behavior is perceived as more natural and human-like, despite its simplicity, owing to the use of social guidance embodied in perceptual cues, to acquire and improve on interactive capabilities from experience. This is significant since human intentions and preferences are determined by latent factors that are manifested in these cues [20]. On the other hand, the study exposes complications of PGRL, such as withholding or degradation of the desired feedback, the inaccurate bases on which the reward function is crafted [97][62], or the use of random policy perturbations whose magnitudes, if inadequately chosen, can hinder gradient estimation and destabilize learning: particularly undesirable outcomes in robotics applications [86][85].

In [40], Guenter et. al. use RL to improve upon constrained reaching movements for goal-directed tasks learned from demonstrations, with particular emphasis on adapting executions to unforeseen situations. The authors use GMMs to encode a 'model' trajectory from multiple demonstrations, trained using standard EM, with timesteps as the input variable and joint velocities as the outputs. This facilitates generalizing trajectories to different goals using GMR, and driving a dynamic system, which is inspired by the human reaching movement model, VITE. While this system can handle reasonable perturbations that necessitate adjustments, these may be limited by what was observed in a set of demonstrations, leading to failures that require either demonstrating what to do in this new situation, or enabling the robot to learn what to do autonomously. The inclusion of RL addresses this problem, introducing some exploration process that would enable learning new solutions to the given task. The algorithm used to facilitate this adaptivity is the gradient-based, value-based Natural Actor-Critic (NAC) algorithm, which continuously monitors the trajectory executed by the dynamical system, stepping in and adjusting parameters through exploratory trials whenever a goal cannot be reached, thus ensuring a novel solution can be attained. For evaluation, the Fujistu HOAP3 robot was taught two tasks: placing an object in a box, and grasping a chess piece, through kinesthetic demonstrations, and its subsequent performance in different task contexts was tested. The first task was impeded by obstacles at different positions along the model trajectory, which could only be avoided when the RL algorithm modified it, while maintaining its shape, to guide the robot through the task successfully. In the second, a chess piece originally could not be grasped from different initial positions of the robot, due to limitations of its morphology, but active exploration of the solution space helped the robot learn new reaching movements. *This presents a strong case for the integration of RL in a LfD approach, which may otherwise be lacking in, if not devoid of, exploration pursuits.*

The utility of connecting imitation and self-improvement strategies in robot skill learning and generalization is similarly highlighted in [63] by Kormushev et. al.. In essence, the authors present a skill representation based on a mixture of dynamical systems, initialized from human demonstrations of a task, which are used to learn couplings across multiple motor control variables. This is essentially a variant of DMPs which incorporates correlation information in its equations. Policy learning by Weighting

Exploration with the Returns (PoWER) [60] (discussed in section 3.1.2), a gradient-free EM-based policy search algorithm, is used to refine and modulate these representations of the demonstrated skill. Among the reasons for this algorithm's use, and its evident popularity, are its independence of gradient estimates and, thus, learning rates, and suitability for learning in high-dimensional spaces. In addition, 'importance sampling' is employed to make the best use of most successful experiences of the robot, reducing the number of roll-outs required to converge, enabling on-line learning. The authors present results of their approach in two experiments: one involving a simulated reaching task that requires obstacle avoidance, and another in which a robotic manipulator is tasked with flipping a pancake on a pan. The reproduction of both is validated, and the high variability of the dynamics in the second task is adequately handled by providing a single demonstration to initialize the RL algorithm, as opposed to relying on multiple demonstrations, for which the variability in trajectories could be a concern. Contextually adaptivity with this approach, however, remains relatively unexplored.

The tuning of motor primitive parameters, using RL, to adapt to new situations is dealt with by Kober et. al. in [61], for which Cost-regularised Kernel Regression (CrKR) is introduced. Aiming to avoid re-learning tasks, the authors use RL to adjust meta-parameters of DMPs, mainly the spatial and temporal scaling parameters, such that behaviors can be generalized in scenarios that require only that level of adaptation. CrKR, which is similar to GPR, enables learning a mapping from situation/task parameters to the meta-parameters, while encoding a predictive variance that implicitly facilitates exploration in an on-policy, non-parametric, policy search algorithm. A group of complex tasks were used to evaluate performance both in simulations and on real robots, including a dart throwing game, table tennis, and ball throwing. A notable limitation here however, particularly from the perspective of the present work, is that more complicated characteristics of trajectories, such as the shape, have not been learned using CrKR, which may limit its potential when generalizing tasks that call for altering the dynamics of the robot's motions in that manner. Moreover, it may share the computational burdens exhibited by the very similar GPR, which occasionally complicates on-line learning, except when certain mitigating steps are taken.

Another use of the PoWER algorithm with DMPs, but in a more unorthodox approach, is presented by Da Silva et. al. in [28], where the familiar problem of learning the mapping between task and policy parameters is posed as the estimation of a topology of a lower-dimensional manifold, from which the appropriate skill parameters can be extracted. The authors assume that different realizations (or contexts) of a task are sampled from some distribution, and acquire task samples and their corresponding solution policies, i.e. demonstrations, to train a family of non-linear regression models that can learn that mapping, enabling the construction of a *parameterized skill*. ISOMAP, a method used to find global geometries of high-dimensional spaces and the number of non-linear DOFs associated with them, is used to extract the topology of the policy space on which the skill parameters lie, which facilitates training the classifier that chooses the particular manifold of the space, and using non-linear regression models (SVMs here) to finally map the context's task parameters to those of the optimal policy. In a simulated dart throwing task, different instances of the task were generated by choosing different dartboard positions, after presenting full arm movements as training examples, then the extracted skill manifolds are analyzed. Interestingly, it is found that the algorithm embeds the 37-dimensional space of policies in a 2-dimensional skill manifold,

signifying the confinement of the policies to a subspace of same dimensionality as the task parameters (having 2 DOFs). Following the manifold selection step, the progressive improvement of the PoWER algorithm of the DMP parameters with experience demonstrates the framework's ability to obtain different 'strategies' required to solve a sampled task. This raises a question regarding the applicability of this skill manifold topology; whether it exists for other scenarios and how much experience is required to reliably extract it. It is also natural to wonder about which task instances to sample for most efficient learning progress, given a task distribution; this is an instance of a curriculum learning problem.

In [57], Kim et. al. focus on improving LfD for cases in which demonstrations are either sub-optimal or scarce, contributing a value-based RL algorithm, Approximate Policy Iteration with Demonstration (APID), that accomplishes that end. Combining expert and interaction data, which is synonymous to the aforementioned apprenticeship learning, is the core concept of the algorithm, which is expected to perform better for real-world policy learning problems, by offsetting the individual drawbacks of each. The intuitive idea, here, is to construct a unified dataset of state-action pairs, containing those collected by executing roll-outs (RL data) and observing demonstrations (expert data), the latter taken as actions sampled by an 'expert policy', and using this to learn the optimal *Q-value function*, from which the optimal policy is inferred (refer to section 3.1.1 for the foundations of value-based RL). As described in the paper, the process is akin to shaping the value function in the subspace spanned by the expert samples, while improving the policy using RL everywhere else. In a simulated car brake control and a real robot path-finding tasks, the authors learn the policy required to choose the correct action from a finite set of primitives, such as move left, move forward, brake, do nothing, etc. The simulated task was tested for varying cases of demonstration sub-optimality and scarcity, against benchmark algorithms. APID was shown to outperform the others for the robot navigation task, even when provided with a single trajectory demonstration, requiring about 10 iterations of RL to converge to some specified goal.

In [70], Levine et. al. achieve autonomous learning, execution and generalization of complex, bi-manual robot manipulation tasks through extracting skills using linear-Gaussian controllers to represent trajectories, and Guided Policy Search (GPS). Sharing some similarity to DMPs, the linear-Gaussian controllers used here could explicitly encode a distribution of actions, like joint torques, in terms of join states at each timestep, and represent any Gaussian trajectory distribution. With a few system interactions, a form of trajectory-centric controller optimization is performed to train multiple controllers in a linear-quadratic-Gaussian (LQG) problem to successively optimize the trajectories required to solve a particular task. It is worth noting that controller optimization updates are subject to constraints bounding the deviation from the current controller in order to limit information loss: a technique very similar to one applied by the REPS algorithm (refer to section 3.1.2 for more details on information-theoretic PS). Although the resulting controllers are capable of executing the task under reasonable perturbations, they may not handle larger variations in task context. Therefore, GPS is used to combine the results of one or more controllers to learn a more general, nonlinear neural network policy, with arbitrary parameterization. Here, GPS involves sampling from the optimized trajectories, provided by the learned controllers, to train a policy in a supervised fashion, instead of training directly using RL (the more conventional approach), resulting in policies that can handle wider ranges of operating conditions than the individual controllers.

Preceding experimental evaluations, the authors emphasize *the importance of defining good cost (or reward) functions that both minimize system interaction time, and obviate manual engineering or tweaking for similar behaviors, particularly for manipulation tasks.* A set of contact-rich tasks: lego-stacking, toy assembly, inserting rings onto pegs, and screwing bottle caps, were used to demonstrate efficient autonomous learning of the controllers, with robustness to minor perturbations, and generalization to different target positions after training neural policies with GPS. Perhaps, the most impressive observation here is the non-requirement of demonstrations, since the robot is able to learn these tasks from independent experience. Despite our focus not being on contact-rich manipulation, this work presents interesting implications on policy formulations, and the fact that appropriate ones may expedite fully autonomous RL, obviating the need for expert demonstrations.

While demonstrations are not guaranteed to be abundant, reliable, or even feasible, it is difficult to overlook their role in policy initialization and accelerating learning in difficult problems, which is why LfD has also been applied in conjunction with deep reinforcement learning. Hester et. al. ([44]) tackle the difficulty of complicated benchmark RL algorithms by leveraging available demonstration data, collected during previous executions by a person/machine that performs as desired, to pre-train an agent before it commences self-improvement using data generated by a deep RL algorithm. This boosts learning by starting with a better initialized policy, especially for cases in which agents must learn in real-world environments, and in which no simulators are available to facilitate acquisition of experience from a huge number of trials, away from the safety concerns of initial, sub-optimal behavior. DQN, the quintessential deep RL algorithm that approximates the Q-value function using a deep neural network, is extended in this paper to the Deep Q-Learning from Demonstrations (DQfD). A pre-training phase in DQfD learns a value function that imitates the demonstrator, using supervised losses and a TD (temporal difference; refer to an overview on concepts of value-based RL in section 3.1.1) loss which ensures it satisfies the Bellman equations so that learning can continue using RL updates from that point on-wards. In experimental comparisons of the algorithm alongside Double DQN (no demonstrations) and plain supervised imitation (no self-improvement), on benchmark Atari games, the virtues of this manifestation of apprenticeship learning in a deep RL scenario are evident. As expected, DfQD outperforms both approaches, doing so even when provided with intentionally poor demonstrations, and managing to learn policies that are more optimal than those of the best available demonstrations. It is important to note however, that the conclusions apply to environments with some similarity to the tested Atari games, with relatively less bearing on complex, dynamic robot manipulation tasks, for example, and those in which contextual adaptivity is of appreciable consequence. Another factor is the use of neural network policies, which may not be the choice solution for other apprenticeship learning frameworks, and thus renders the assumption that demonstration data and reward signals can be straightforwardly combined inadmissible.

In [42], Hazara et. al. utilize and improve upon the aforementioned GPDMP LfD model ([71]) in an RL-based incremental learning procedure, taking a principled approach to incrementally constructing a database of motion primitives, with the objective of better generalization to new situations in robotic tasks. Learning a skill in environments exhibiting regular perturbations is framed as an incremental learning process, in which an ideal trajectory, which must be followed using RL, must first be estimated for

each new situation. The authors postulate that the uncertainty when predicting the mapping between a task parameter and corresponding motion primitive parameters should be utilized to enhance the learning process. GPDMP, which generalizes imitated tasks to unseen situations by learning this mapping, is a non-linear global parametric LfD model that was previously shown to outperform local and global linear models, particularly in extrapolation capability and computational savings. In this paper, it is extended to provide estimates of prediction uncertainty, similar to what is provided by the more expensive GPR, with which a policy search-based RL exploration process is guided. Having a database of DMPs, GPDMP extracts the underlying distribution that is used to generate DMP predictions for task instances. If unsuccessful, these predictions are optimized iteratively using the model-free PoWER algorithm, and are otherwise added to the database, updating the underlying GPDMP model. The exploration, and thus successful convergence, of the RL algorithm is driven by the covariance of the DMPs stored in the database, which is learned using an empirical Bayes approach, similar to that used in covariance matrix adaptation (CMA). The result is an incremental, online procedure that iteratively builds a general motion primitive generative model, whose performance is demonstrated with the ball-in-a-cup task on a real robot, the KUKA LBR 4+. The intricate framework, while fairly complex, possesses qualities conducive to sustainable. autonomous skill acquisition, including requiring a single expert demonstration, using experience data efficiently, and exploiting predictive uncertainty to systematically guide exploration. Similar to adjacent methods, however, the nature of context variations that can be encoded and generalized across by the acquired skill remains to be thoroughly investigated.

A more lucid implementation of contextualized action, or skill, adaptation by Colome et. al. can be found in [27], where the best actions for different contexts in some robot manipulation tasks is learned in an approach that employs probabilistic movement primitives (ProMPs) and GMR. A notable resemblance to the present work is in the representation of changing environmental conditions in the form of dedicated context variables, with the objective of generalizing learned MPs to various task contexts. Mentioning the lack of scalability in frameworks that would aim to describe what action to perform in every context, due to the curse of dimensionality, the authors attempt to build a generative model that learns common contextual features of similar actions that can perform a task in a situation, and thus generate appropriate ones, given the context. Since ProMPs traditionally require a large number of parameters to encode several actions that have common context features, the authors opt for first applying a successful EM-based dimensionality reduction technique. The more compact representation is used to learn a GMM that also includes a context variable, where the authors use *Persistent Homology* to determine the optimal number of Gaussian components. Subsequently, given an observed context variable, a conditioned mixture model is used to sample the appropriate robot commands, for example, that best accomplish the task; GMR is thus used to generate the most likely MP/action parameters, given the context. The relation of this approach to RL lies in a clever model learning method: the GMM model is updated using reward-weighted, performance-specific component responsibilities, with these rewards being converted to relative weights with a policy search algorithm such as REPS. The resulting EM updates of the model then closely resemble a direct RL algorithm. In a mannequin-feeding and peg insertion robot manipulation tasks, the authors experimented with the generalization capability of their model. Different

object positions and types requiring adjusted reproductions for the feeding task were successfully adapted to, following initialization with 40 demonstrations. It is conceivable that the probabilistic nature of this model, especially owing to the use of ProMPs, introduces a dependence of the generalization performance on the distribution of demonstrated trajectories, evident from the authors' own care to include as much contextual variation in them for their experiments. Nevertheless, the dimensionality reduction of the MP parameter space is a noteworthy contribution of the work, among others, which may prove to be a necessity for problems with high-dimensional, continuous action spaces.

We now turn our attention to two particularly relevant classes of RL algorithms: hierarchical policy search and contextual policy search (refer to section 3.1.2 for policy search formalisms).

Hierarchical Policy Search

The term hierarchical reinforcement learning (HRL), due to some obscurity, may refer to one of several concepts, including applying RL at the highest level of a control system hierarchy, generating skill hierarchies in a developmental approach through RL, or learning policy parameters of sequences and multiple layers of policies, simultaneously [101]. Strictly speaking, the first does not qualify as HRL, while the last presents the most recognizable form of the term: the introduction of temporal abstraction to RL problems, enabling the construction of a hierarchy of policies, usually called options, which determine the selection of lower-level options, or 'leaf' action primitives. This can serve to exploit the hierarchical nature of tasks (such as cooking), or avoid learning a single 'monolithic' policy in favor of effectively introducing decision points along a policy that may help with learning both the optimal choice of behaviors or actions in a given situation, and the optimal parameterizations of each choice. This is highly suitable for the contextually adaptive skill acquisition we seek, as expressed in the reviewed literature.

Stulp et. al. in [101] present an interesting framework for integrating DMPs into a hierarchical RL approach for learning sequential robot skills, using the policy-based Policy Improvement through Path Integrals algorithm, PI^2 ([106]). The concepts of task decomposition and temporal abstraction are highlighted as fundamental for reducing search spaces and making complex tasks, particularly those with underlying action hierarchies, more manageable to learn. Here, HRL is applied for optimizing sequences of DMPs, such as for segmented pick-and-place tasks, by following a method very similar to our own: learning the trajectory shapes and goals to solve the task optimally. The hierarchy here refers to the simultaneous learning of both parameters, at different levels of temporal abstraction, using a unified cost function and update rule. Similar to the present work, they initialize their policies (the DMPs) with demonstrations, then use PI^2 to learn the parameter exploration parameters, which are directly encoded in the DMP equations, and which allow sampling roll-outs for learning. The author's observations of human executions in a pick-and-place task yield interesting insights: subjects' theoretical motion primitive 'parameters' were influenced by those required for subsequent primitives, and intermediate goals were optimized with respect to 'cumulative cost' of the entire sequence. These notions principally informed their policy update strategy: within a sequence of DMPs, the shape parameters are updated according to the "cost-to-go within a primitive", while goal parameters are updated according to total cost of the current and all subsequent primitives. In an experimental setup, results from a robot manipulator tasked

with a similar problem demonstrated the superiority of this HRL method over learning only shape or goal parameters, or using naive cost updates that do not consider the whole sequence of primitives. This work shows the merit of learning DMP trajectory shapes and goals hierarchically, as we do in the present work, even if the task itself is not necessarily hierarchical in structure.

Another hierarchical approach to a policy search algorithm by Daniel et. al. extends the basic version of the one we employ in this project, REPS (refer to section 3.1.2 for a brief explanation, and 5.1 for a detailed analysis), to a hierarchical variant named HiREPS, which applies the prevalent mixed-option policy formulation [29]. In it, a gating ('supervisory') policy picks *options* which are separate policies analogous to movement primitives or templates, given the state, and each option (sub-)policy describes the behavior under that option, specifying the eventual low-level actions to be executed. As previously mentioned, this avoids a single, 'concentrated' policy, and the authors utilize it to learn multiple solutions for a single task, if they exist, such as reaching for an occluded object using any of multiple redundant paths. REPS, an information-theoretic policy search algorithm, is formulated to incorporate this structure: the optimal hierarchy is initially unknown and the option variable, o, unobservable, for which an EM algorithm is used to estimate these latent variables, creating a 'proposal distribution' from samples of the current policy at each time-step, and subsequently updating both classes of policies using reward-weighted MLE. The exact details of the algorithm are omitted here, for brevity. It is worth noting, however, that the authors add a novel constraint to REPS's constrained optimization problem that ensures options are clearly segregated in the solution space, avoiding undesirable overlaps. Presumably for simplicity, the paper concentrates on episodic cases, in which a parameterized option policy, once selected by the gating policy, is executed until the end of an episode, much like what is done in the present work. Using some simulated trajectory-centric robot tasks, which call for deciding between equally satisfying solutions to the same problem, the authors show the HiREPS extracts the underlying decision hierarchy, outperforming REPS, in the designated tasks. An intriguing property of HiREPS is its handling of problems with multi-modal solution spaces, which deceive most PS algorithms into averaging over the optima of each mode, thus converging too slowly, or to a decidedly sub-optimal solution (imagine driving straight into a tree as a result of the indecision between to equally suitable options: swerving left or right). Exploiting the hierarchical policy structure, HiREPS could successfully identify both solutions and segregate them, strictly choosing either. The algorithm has since been evaluated on a real Barrett WAM arm, particularly demonstrating generalization of a motor skill by learning concurrent solutions to a task [30], and more extensively developed and analysed in a long version of the paper, [31].

More recently, Pinsler et. al. ([91]) proposed a HRL framework that incorporates human feedback in context of robotic grasp skill learning. Since designing robust reward functions for intricate grasping tasks is known to be far from trivial, they opt for learning a bi-perspective reward that combines models from both human feedback and the robot's predictions, in a way that maximizes feedback and sample efficiency, respectively. In essence, preference feedback on task outcomes are collected from a human to learn an outcome reward model (ORM), on the low-dimensional 'outcome space', and the resulting information is consolidated with the robot's learned context-parameter reward model (CRM), which predicts a chosen parameter set's performance prior to a roll-out, by back-propagating it to this higher-dimensional policy

parameter space. Both models are learned using GPs. This is used in an HRL scheme that relies on contextualized policies, so that complex grasping tasks are realized through more generalizable motor skills. Here, an upper-level policy $\pi(k|s)$ determines the context-parameter pair best describing a given problem (where k represents an option, c: a context), basically choosing a low-level option policy $\pi(\omega|c)$, which in turn generates the appropriate motion parameters, given the context. The upper-level policy relies on the contextual GP-UCB acquisition function to greedily choose the best option, based on the GP reward model, and the lower-level policies are simply Gaussian distribution that are individually updated using the REPS algorithm. The presented HRL with Bi-Perspective Reward Learning from Preferences algorithm observes the current context c, drawn from some distribution $\mu(c)$, selects the most promising option policy using the upper-level policy, based on the learned CRM, performs a roll-out by sampling motion parameters ω from the lower-level policy, initialized from demonstrations, and decides whether to ask for human feedback based on the expected change in the distribution of the ORM. Finally, the reward models and lower-level policies are updated. Since this bears a striking resemblance to the C-REPS algorithm we apply in our work, it is important to point out a few differences. In C-REPS, the upper-level policy $\pi(\omega|c)$ chooses lower-policy parameters given the context, while the lower-level policy $\pi(u|x,\omega)$, generates low-level actions. The former is implemented as a linear-Gaussian model, as opposed to their GP model; the latter, as a DMP, instead of simple Gaussian models. Crucially, the learning is performed using REPS in either case, but is done at the upper level in our case, and at the lower level in theirs. These differences make our selected approach more adequate for the addressed problem. On the other hand, the integration of a second form of feedback from the human, which is absent in our case, may help accelerate learning, as is demonstrated in the authors' grasping tasks, subject to different object types. Finally, a model-based extension of the algorithm has not been introduced, which could have been of greater significance to the present work.

In conclusion, the relevance of a hierarchical view of RL to our problem lies in making policies amenable to contextualization, and thus better suited for generalization of robot skills. Since a reasonably compact, monolithic policy, however expressive, would struggle to capture optimal behavior in multi-modal and context-dependent solution spaces, this seems to be an intuitive extension. Apart from ensuring feasible learning in some problems, it has also been shown to increase sample efficiency [91], which directly follows from these facts. As discussed, the inclusion of dedicated context variables has already been applied in HRL frameworks, and approaches that characteristically do so are discussed in the next section.

Contextual Policy Search

CPS extends PS algorithms to allow contextual learning, i.e. learning in cases where solution spaces are not necessarily uni-modal, owing to the different situations that characterize the nature of a task, and where policies must cope with variable operating conditions, on which optimal behavior may significantly depend. In these cases, an agent chooses policy parameters based on the current observed context variable, presumably drawn from some stationary distribution, and learns from a context-dependent reward signal. A context variable sufficiently specifies the particular setting of the task, to which the policy must adhere. CPS algorithms, although not necessarily hierarchical, are usually formulated in hierarchies due to the

aforementioned motivations. This section contains a brief review of some CPS approaches, the first of which we have determined to be particularly suitable for our problem.

In [67], a contextual variant of the model-free Relative Entropy Policy Search (REPS) algorithm, C-REPS, is introduced by Kupcsik et. al. to deal with the learning and generalization of robot skills. By augmenting the algorithm with GP forward models that capture the dynamics of the robot and its environment, at least pertaining to the task at hand, they create Gaussian Process Relative Entropy Policy Search (GPREPS): a model-based CPS algorithm that is able to learn higher-quality policies in a more sample-efficient manner thanks to these models, whose predictions substitute real robot trials. GPREPS is further developed and evaluated in [66]. The idea, which has been briefly mentioned, is to have a hierarchy consisting of an upper-level policy, $\pi(\omega|c)$: a linear-Gaussian model that chooses parameters ω to maximize return given context c, and which is learned using REPS, and a lower-level policy, $\pi(u|x,\omega)$: predominantly DMPs, or generally any policy that specifies control actions u (elsewhere also denoted by a) given the current state and the policy parameters dictated by the former. The sample-efficiency previously alluded to is realized by incorporating forward models: initial roll-outs are performed on a real robot to capture the dynamics of the task in GP models, which are then used to sample 'artificial' trajectories and their expected returns that constitute the dataset with which upper-level policy updates are performed, through weighted MLE. These GP models are also refined at each iteration, mitigating effects of model error. The advantages are several: relying less on real robot interactions implies less safety concerns, tedious procedures, wear and tear, and learning time. Additionally, a characteristic problem of model-free RL: high variance in reward, is avoided, and GPs inherently reduce model errors, since they naturally express learned model uncertainty. The hypothesized gain in performance, especially in reducing required interaction experience, is evaluated in comparison to the REPS and PILCO [34] algorithms, in simulated pendulum balancing, ball throwing, and hockey tasks, the latter also tested on a real manipulator, in which different target positions represented task 'contexts'. All experiments proved the superior efficiency of GPREPS, which makes it theoretically more applicable to learning contextual policies in real-robot tasks. This model-based improvement does not come without a price, however, which is evident in the fact that task dynamics being adequately representable using GPs and the infamous computational burden of GPR become notable concerns.

In [2] and [1], Abdolmaleki et. al. improve upon the contextual approach to policy search, there termed contextual stochastic search, following the identification of some drawbacks of REPS. With the same aim of enabling an agent to choose parameters that fit task context, without having to re-learn a skill, the authors find that search distributions (policies) learned using REPS tend to converge prematurely. This is an effect of the weighted MLE used to estimate the covariance matrix, which leads to over-fitting and diminishing variance/exploration too fast. Therefore, they implement a novel covariance regularization method to obtain better estimates, in which a weighted average of covariance estimates over all iterations is used, and the influence of the initial distribution is decreased at each iteration (entropy reduction), giving rise to the Covariance Estimation with Controlled Entropy Reduction (CECER) algorithm. A contextual variation of CECER is identical to C-REPS in every other regard. Results show both contextual CECER outperforming C-REPS in benchmark problems, the latter suffering from premature convergence.

The same authors further extend CECER to introduce two other improvements over REPS. The first addresses a limitation due to the parametric nature of the upper-level policy, a linear-Gaussian model, whose linearity with respect to the defined context variable, c, restricts it to mere linear generalization over tasks. Moreover, while mean is context-dependent, the covariance matrix is fixed (common) for all contexts, not qualifying the search distribution as fully context-dependent. The non-parametric local CECER algorithm, presented in [3], utilizes local Gaussian distributions for the policy, and modifies policy updates such that the covariance matrix is as context-dependent as the mean function, by adjusting the original equations used in REPS. Preliminary results gathered from simulated robot toy task experiments demonstrate the algorithm learning policies that are non-linear functions of context variables, unlike REPS, and assigning different variances to different context values, unlike an older non-linear extension, RBF-CECER.

Naturally, higher-level and more expressive variables have been exploited to represent contexts, instead of simple integers or real-valued vectors, including high-dimensional ones such as camera images. These are traditionally projected into lower-dimensional subspaces using methods like PCA, both to enable applying the aforementioned CPS algorithms and making learning more feasible. The work presented in [105] by Tangkaratt et. al. is motivated by the relative inefficiency of this, however, and contributes a CPS algorithm that incorporates this otherwise pre-processing step. Linear dimensionality reduction techniques, when used for this purpose, may suffer from 'distractor' dimensions: components of the latent representation of the context space which, while they may vary significantly, may have little to no bearing on the achieved reward. Contextual Model-based Relative Entropy Stochastic Search (C-MORE) implicitly performs dimensionality reduction by learning a low-rank representation of the reward function/model, shown to produce an equivalent result. This is learned using a nuclear norm minimization approach that can enforce this low-rank representation. The C-MORE algorithm is very similar to C-REPS, relying on the same linear-Gaussian policy and information-theoretic policy updates. Similar to GPREPS, C-MORE is model-based, but with an interesting distinction: instead of the implied model being used to approximate system dynamics and generate artificial roll-outs, it is used to approximate a reward function. While C-MORE outperforms C-REPS with PCA for tasks whose contexts are identified from raw images, it is only proven to do so in such high-dimensional scenarios, keeping in mind that the nuclear norm approach is computationally expensive and whose use must thus be justified. An unfortunate consequence of relying on locally fitted Gaussians to achieve a globally non-linear model is the fact that all training data must be stored in memory. Nevertheless, the approach offers interesting avenues for more complex context representations, possibly learned from visual perception, and more adventurous reward model formulations, such as deep neural networks.

The use of such non-parametric models for contextualized policies has recently become a recurrent theme in CPS research due to the restricting assumptions of parametric models. While some search distributions, i.e. contextual policies, have means that are linear in context features, or are mixtures of linear models, other problems have more complex mappings from contexts to policy parameters, necessitating more flexible models. In [12], Barbaros et. al. categorize promising non-parametric methods applicable to contextual stochastic search into memory-based (lazy) and eager learning methods, and

investigate their defining features, contributing an algorithm that aims to consolidate their strong-points. Local CECER ([3]) is an example of a locally-weighted, memory-based algorithm, where locally linear models are fit to nearby points to make a prediction, producing highly non-linear global models, but requiring storage of all training data. On the other hand, eager methods include GPs and GPR which escape linear boundaries by computing inner products of the provided features (context variables, in our case), in some kernel space. In their comparisons, the authors conclude that memory-based approaches are hindered by expensive calculations at each prediction, while eager approaches struggle to find global parameters that adequately fit the data in all parts of the search space; drawbacks that are offset by the opposite approach. They thus propose a novel hybrid algorithm that combines the advantages of either (namely, the NP-REPS and local CECER algorithms) which is shown to learn faster and converge on better policies in context-dependent simulated tasks. These included a planar robot hole-reaching task with varied hole positions, widths, and depths, and a cart-pole balancing task with uniformly sampled cart masses and pole lengths.

While we can demonstrate motor skills to robots and equip them with the tools for self-improvement through acquired experience using apprenticeship learning, and even extract skill hierarchies to some extent using HRL, contextualized policies enable behaviour that is flexible to changing contexts. As a result, honed motor skills can become even more general and robust to dynamic tasks and environments. CPS is therefore a promising solution, and the various algorithms studied here, along with their recently proposed extensions, merit further study and evaluation on real robot platforms performing everyday tasks that require contextual adaptivity.

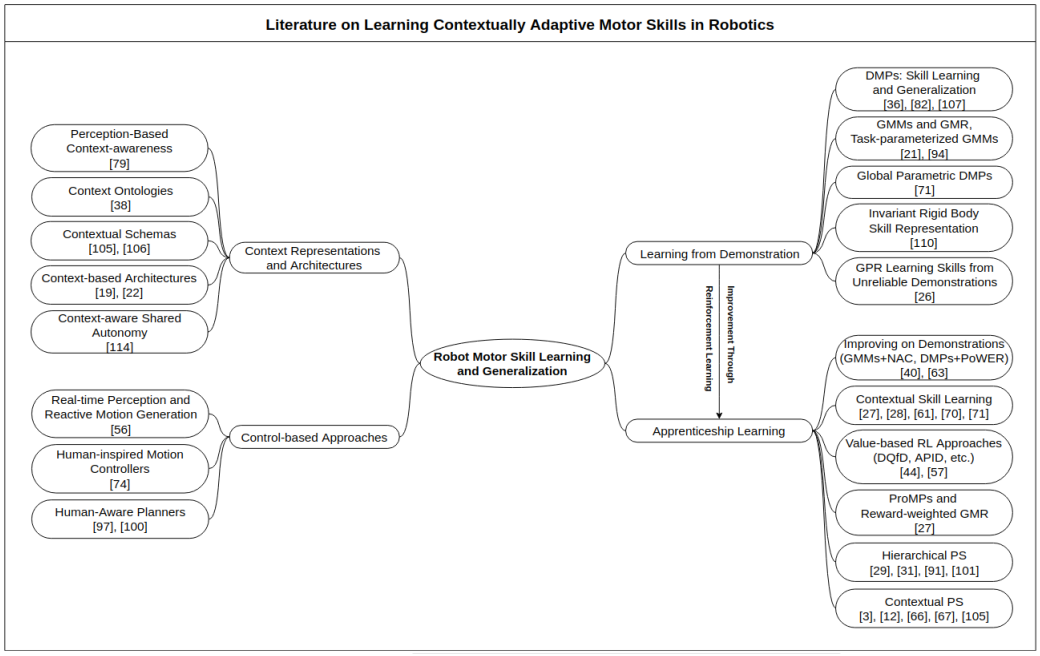


Figure 2.1: Mind-map of the surveyed literature concerning approaches to robot contextual motor skill learning.

2.2 Human-Robot Object Hand-overs

This section reviews some of the research concerning our representative use case for a generalizable robot motor skill: human-robot object hand-overs (HOs). The aim is to gather insights on the nature of the task from the perspectives of both human and robot executions, viable formulations of the skill, principle contextual factors, and how they were adapted to in the past. A collection of enlightening studies is thus briefly surveyed below, summarizing the salient findings and contributions of each. Despite a significant body of research on the topic, it is recognized that human HOs are not understood well enough to allow transferring the latent mechanisms of the skill to robots, suggesting room for further research on the topic.

Kajikawa et. al. and Shibata et. al. present one of the earlier studies on human-robot object HOs in [55] and [98], particularly concerning motion planning, citing the task as an exemplary embodiment of a collaborative everyday action to be performed by a service robot: a sentiment we share and which motivates our choice of it as use-case for our work. In their work, they attempt to study and analyze human HOs, particularly focusing on the characteristics of human motions, such that robot hand-overs are designed to be more smooth, predictable, natural, and human-like. This is done in a qualitative approach: extracting these characteristics and attempting to reproduce similar trajectories, avoiding explicit knowledge of the complex inner systems that produce this behavior in humans. Simple human HO experiments, where objects are transferred in a 2D plane (a table), were used to record and analyze executed trajectories and velocity profiles. The result is a division of the task into phases, and some observed patterns, such as the receiving person tending to commence motion just after the deliverer reaches their maximum approach velocity. The findings are then implemented in a potential field-based path planner, in which velocity and acceleration equations are written to reproduce similar behavior. A simulated robot arm, completely driven by the potential fields, is then shown to follow objectively human-like trajectory profiles, at least within the similarity measures considered in the study.

Human-robot HOs, however, should not solely depend on goal configurations and motion feasibility, with no regard for how the task is perceived by a human. In [58], Koay et. al. express the importance of adaptive, socially acceptable service robot behavior, manifested in task executions, and which motivates their design of a holistic human-aware planner. The paper presents results from a robot hand-over study that analyzed human preferences for factors including robot approach distance, approach direction, and hand-over position/distance, as a robot handed seated subjects an object, with the aim of extracting actionable insights. Experiments were preceded by establishing participant preferences for the aforementioned parameters to be incorporated by the robot, aiming to interactively guide its chosen HO 'gesture'. Four variants of arm-base coordination approach styles were executed by the robot for each participant, tailoring the execution to their said preferences. In addition, questionnaires were used to determine the subjects' personality traits. The authors arrived at some notable conclusions, such as people mostly preferring being approached from the front or right-front, although directions generally correlated with preferred hand-over positions. Interestingly, results on approach distance, once matched with personality scores, implied that 'more agreeable and open' people preferred closer HO interactions, and vice-versa. Comparisons to earlier studies conducted by the group also revealed noticeable effects of cohabituation between individuals

and the robot on their preferences, making experience with robots an appreciable factor. An interesting conclusion, particularly pertaining to the present work, is the fact that participants preferred hand-over positions lower, towards their chests, while they were seated, which proves human posture is a consequential factor, and a valid context variable for robot-to-human HOs. Overall, the results are a strong indicator of the contextual nature of the task, since different executions may be optimal in different situations, and the importance of adapting to human preferences. The findings were eventually incorporated in the human-aware motion planner ([99]), referred to in section 2.1.2, contributing to the legibility, safety, and predictability of robot motions as it executed HOs, which were more thoroughly validated in [32].

In [48], Huber et. al. highlighted a more fundamental aspect of HOs and a focal theme in the concerned literature: minimum-jerk profiles. While the previous paper focused on higher-level attributes such as hand-over position and approach distance, the brief study presented in this work evaluated the trajectory characteristics of a robot's HO, particularly pertaining to its similarity to biological motion. The authors similarly believe that more efficient joint and collaborative actions can be achieved by transferring knowledge gained from human-human interactions to robots, prompting the need for such intricate studies of human HOs. Comparisons were thus made from human-human HOs and human-robot HOs executions in an identical task, to draw conclusions on two velocity profiles: trapezoidal, calculated in joint coordinates, and minimum-jerk, which is based on a model with an objective function that ensures smoothness of the end-effector's trajectory. Variables like HO duration and reaction, manipulation, and post-hand-over times were recorded along with trajectories during human-human trials, and compared to those of human-robot trials. Although trapezoidal profiles are the norm in conventional arm control, the authors concluded that the more natural minimum-jerk profiles were perceived as safer, and seemed to decrease task reaction times, possibly due to their predictability. The results show how contemporary robot technology can be moderately adjusted to produce surprisingly more human-friendly and efficient motions, taking inspiration from human task executions.

Dehais et. al. attempted [32] to define more objective metrics with which to evaluate trajectories produced by planners for human-robot HOs, following studies that deemed subjective questionnaires more informative than oft-used quantitative criteria, such as human reaction times or accuracy. They thus opt to employ both qualitative data from participant accounts and ratings describing their perception of the robot executing the task, and physiological data comprised of skin conductance, deltoid muscle activity (EMG), and ocular activity. The HOs were planned using the human-aware manipulation planner ([99]), which computes HO trajectories based on criteria of legibility, safety, and the person's physical comfort, with the actual aim of this study being the evaluation of these metrics as tools to be used for optimizing these criteria. In a thorough experiment, the authors design three motion types leading to different trajectory characteristics of the Jido robot's HO, one using the planner and grasp detection with a medium velocity setting, a second using neither and with a high velocity setting, and a third using the planner, no grasp detection, and low velocities. Subjects graded the first motion the highest for legibility, safety, and comfort, proving general preference for trajectories resulting from the planner's empirically defined criteria. Additionally, it was found to elicit the lowest muscular activity, while the second caused highest skin conductance measurements, implying surprise or stress due to the sudden movements, and

also caused higher ocular activity, along with the third. This work illustrates the effects of different motion trajectories on humans' perceptions and physiological responses, and how factors that govern these should be taken into account, further lending credence to the idea that robot motor skills should be adaptive to the humans involved in the task.

In [20], Cakmak et. al. similarly pursue the objective of more 'natural' and 'appropriate' HOs in a framework that more directly captures humans' preferences, by 'learning' from examples, in a sense. The incorporation of these preferences is sought to enable more seamless human-robot HOs, particularly regarding HO *configurations*. Here, configurations are specified by grasp pose relative to the object, arm configuration, and robot position relative to a person, as they execute a HO task. The authors gather data from human-robot experiments to learn preferred configurations, then evaluate how they compare to those obtained using a planner that chooses configurations based on a kinematic model of a person, such as that of [102] (discussed in section 2.1.2). The latter approach takes human kinematics into consideration, solving a hierarchical optimization problem to constrain and solve for the different HO configuration variables, producing what the cost/value functions think is the configuration that best reflects kinematic attributes of the human. OpenRAVE was used for simulation and grasp planning. In order to learn configurations from users, for the other approach, preferences were encoded in the robot's decision process by letting users choose parameter values for good and bad examples of HOs, through a GUI, which are later used to discriminate between enumerated configurations and to choose the one most likely to be preferred. A systematic experiment in which HERB, a service robot equipped with two 7-DOF WAM arms and 4-DOF Barrett hands, was performed to compare the purely 'planned' configurations with 'learned' ones. The results show that kinematic planner chose configurations that may be considered 'practical', but the learned solutions proved more usable, natural and appropriate, and generally preferred, according to user evaluations, since they implicitly encoded these attributes. Intuitively, and as this study shows, leveraging learning directly from human preferences tends to improve human-oriented skills, such as HOs, and is an important reason for our use of LfD. We should note, however, in the words of the authors: "... that the hand-over configuration is one of the many factors that influence hand-overs. A complete hand-over behavior will need to consider other factors such as the robot's trajectory or the person's posture and gaze direction".

In the interest of objectively comfortable robot-to-human HOs, Aleotti et. al. more extensively plan object-dependent HOs in [7] by adjusting object orientation to suit human preferences. To that end, a planning-based approach is employed in which HO configurations are chosen to be most comfortable for a user, by delivering an object in an orientation that best suits its reception by the person. In this perception-driven and more holistic approach to the task, a laser scanner and Kinect are used to construct a 3D model (point cloud) of an object and detect a user, synthesizing a motion planning simulation environment. A planner implemented using the OpenRAVE engine then determines a grasp and end-configuration for the object that aims to induce the least effort on the user, when they receive the object. Experiments conducted with the Comau SMART SiX manipulator utilizing this planner concluded that resulting HOs were subjectively deemed comfortable. Although no objective means were used to validate these conclusions, the study provides some insight on planning HOs of unseen objects at a larger

scale, for which perception elements may be key when it comes to adapting for object characteristics.

In the same vein of research exploring human-human HOs and considering wider perspectives of the task to better ground human-robot HOs, Strabala et. al. investigate and attempt to emulate the physical and social-cognitive aspects of the task, on robots. In [100], they conduct a study that consolidates past knowledge on HOs to design a basic structure, guided by insights from human evaluations and observations from implementations, to formalize and codify a common procedure whose results can come closer to mimicking the way humans achieve seamless coordination when performing the task. The implied spatial and temporal coordination includes the major phases of approaching, reaching out, and transferring an object. An initial handover structure was extracted from human executions, providing insights on human intent establishment and signaling. Participants handed different objects in varying contexts over to each other, and video data was subsequently used to meticulously analyze each phase of the task, leading to various interesting insights indicating the complex social-cognitive channel involved when implicitly signaling transitions of the task to the other person, apart from physical channels. A separate experiment involving HOs in a collaborative context confirmed that humans communicate hand-over intent and coordinate reaching action initiation through prior implicit signaling. Crucially, the authors highlight context: the state of the world before commencing the HO, as a vital influencing factor. This work later lead the authors to some user-centered designs for robot HO behavior, which were evaluated on a series of studies that explored how robots can establish the 'what, when, and where' details, and consequently achieve more seamless executions, but the results presented in this paper remain seminal in HO research, informing various other studies. Evidently, the authors were the first to conduct studies wholly focusing on human-to-robot hand-overs, as opposed to the opposite, and contribute human-adaptive design rules that can be readily integrated in conventional motion planners employed for HOs: a noteworthy contribution to collaborative robot design.

An adjacent procedure is followed by Chan et. al. in [23], where they introduce the idea of the object-specific affordance axis: an axis that givers orient an object with respect to, given its properties and the manner in which a receiver would prefer it to be handed over. These are extracted from mean orientations of different objects handed over in different contexts by human subjects, with the aim of teaching robots to execute the HO task with particular objects' affordances in mind, promoting better collaborative behavior. The study similarly offers rich insights on human performance of HOs, and underlying, largely unexplored dynamics.

In [39], similar 'joint action' signals identifying human engagement in a HO task are utilized by Grigore et. al. with the aim of increasing adaptivity of the HO skill, which overlaps with our objective of generalization over different contexts. The robot-to-human HOs studied here were in settings involving a person whose attention and readiness to receive a requested object varies, due to some cognitive load, requiring a robot to robustly estimate the states of the task, and recognize the intentions of a person through non-verbal cues, avoiding blind, failure-prone executions. Hidden Markov Models (HMMs) are used to model and estimate the sequential states of a HO, after learning them from task observations, capturing basic hand-over dynamics in this first, coarse layer. The principle of joint action understanding, derived from human executions, motivates a second 'cognitive' layer that monitors a human's focus

through eye-head gaze orientations, subsequently informing the robot's final decision of when to release an object, in the final HO state. Analyses of experiments where a BERT2 robot and a VICON motion capture system were used to performs HOs using this framework showed that more successful and 'safe' HOs were realized by integrating non-verbal human co-worker cues, as opposed to rigid executions that only rely on physical aspects of the task. Undoubtedly, sole use of human gaze direction may lead to intent predictions that are inaccurate, considering their deceptive nature and possible variance across people. However, utilizing human social cues like these is a step towards more adaptive, less rigid executions of context-dependent HOs, whose overall flow will be dictated by the given situation and state of a person.

Other interesting adaptive task coordination strategies for human-robot HOs were investigated by Huang et. al. in [46], who observe human temporal coordination and co-adaptation in HO tasks, particularly pertaining to partner workload, and implement similar behaviors on a robot manipulator. In tasks involving multiple hand-overs, such as the presented dyadic dish unloading task, said behaviors involve an awareness of the task status of the collaborating person, and consist of an adjustment in movement speed and inter-action waiting times. These were derived from human demonstrations of the task, from which the authors noticed that an object giver monitored a receiver's progress as they stowed away a received object, and possibly attempted to achieve an unrelated objective, appropriately slowing down or pausing their activity to adapt to their partner's workload. Processed time-series data of human joint readings, extracted from raw video, allowed the identification of patterns leading to the discovery of these two strategies, which were used interchangeably: pausing an action or slowing down, while the other person is not ready. Transferring this to an autonomous robot, the authors set up a Kinect sensor to extract joint data from a person collaborating with a robot in the same task, in order to estimate their state from body configuration,. Additionally, certain waiting and slowing behaviors in the robot's actions were encoded in an algorithm that receives the predicted user state and adapts to them using these behaviors. In experiments, this 'adaptive' strategy was compared to a proactive strategy and a reactive one. When acting proactively, the robot would fetch a second object in a sequence as soon as it delivered the first, presumably minimizing idle time to maximize team performance. In the reactive strategy, it would wait for the person to be available again before it reached for the second object. The main conclusion of the study was that adaptive strategies seem to balance an inherent trade-off between team performance and user experience best, offsetting drawbacks of the extreme alternatives. This implies that sole maximization of task performance in human-robot collaborations may not necessarily result in most the desirable joint action, due to novel constraints imposed by the inclusion of humans, for whom contextually adaptive behavior is naturally ideal.

In the work of Quispe et. al. in [92], user preferences for hand-overs given their location and current activity are learned in a probabilistic model, such that a range of conceived contexts are adapted for. An initial survey was used to justify the need for multiple HO 'policies', by asking users about various situations and how they would grade example HOs performed by a robot, from among simple execution 'styles' (robot approaching the user, vice-versa, and robot placing the object on a nearby table). The results proved a distinctive dependence of preferences on activity and location, and were used to codify four hand-over 'policies', depending on both factors. A simple 3-node Bayesian Network (BN) is then

used to learn the preferred policy for situations with five different activities (walking, eating, etc.) and locations (corridor, office, etc.) each. At each execution, the robot would observe the 'context', pick the hand-over policy with the highest likelihood, and receive feedback with which the BN's weights are updated using an EM-like approach. The authors show that around 50 interactions are sufficient for a HSR to learn the policy preferences fairly accurately. While these may dependent on various factors other than just the two considered here, the authors point out that depending on how a robot is used, even this simple behavior may be adequate. This idea has implications on how our concept of contexts ie embodied, including representation and granularity. An updated version of this work is presented by Martinson et. al. in [72], in which the user-specified considerations for hand-overs extracted from the survey, are discussed in more detail. It is worth noting that human posture, specifically sitting versus standing, was rated the third most important factor to consider for adequate hand-overs, valued almost as much as location, and more than user feedback, object (type), and safety.

More recent studies on HOs confirm the continuing efforts to gain more knowledge about the human aspects of this seemingly simple task. In one such study by Rasch et. al. in [94], the authors attempt to move closer to human-like robot HOs, in the hopes of implementing controllers for humanoids and service robots that can execute similar movements. An extensive user study was conducted to investigate human subjects' movements, including shoulder, elbow, arm, and wrist movements, driven by the hypothesis that trajectories executed by people handing objects over follow some pattern, and not simply shortest paths, for example. Interesting insights were gathered following the analysis of IMU sensor data and videos collected from experiments, the primary being the discovery of an underlying general pattern of four movements that are generally executed sequentially by an object giver to accomplish a HO, including rotating the shoulder, flexing the elbow, and so on. With these data, a novel trajectory motion model for human-robot HOs was developed to incorporate the shape of an exemplary HO trajectory, partly based on the minimum-jerk model. However, since the human-like trajectory curved in space along all axes, neither a linear nor the minimum-jerk model was adequate, with the authors opting instead for a decoupled five-degree polynomial. Overall, the authors' work contributes to the growing body of research aiming to enhance user acceptance of and comfort with service robot executions of everyday tasks, by pursuing movements that resemble the ones the users would execute themselves.

In conclusion, the latent intricacies of the human hand-over skill are apparent from the results of studies conducted for the reviewed works, which reveal non-trivial challenges of transferring the skill to robots, not least of which is the issue of natural and preferred behaviours. Whether to attempt to emulate strategies and techniques observed in human executions or pursue novel formulations for hand-overs that better adhere to a robot's capabilities remains an open question, but the attempts documented in the literature provide useful illustrations of the possibilities, and a good starting point.

3

Preliminaries

3.1 Reinforcement Learning

Reinforcement learning (RL), classified as the third machine learning paradigm, alongside supervised and unsupervised learning, is a process by which an agent learns optimal behaviour from experience through autonomous interactions in an environment in which it is situated. Given no directions on actions it must take, the agent attempts to learn this mapping from observed states to actions by maximizing a scalar reward signal, the only indication of its performance provided by the environment, which is used to progressively discover the best action(s) to take in a given situation.

RL suits the problem of autonomous robot learning well: interaction-driven, iterative inference of optimal behaviour in a dynamic, partially observable environment, drawing on inspiration from biological processes. We, as humans, acquire experiences ranging from primal instincts to complex feats of coordination such as walking, solely through trial-and-error in a life-long learning process. This ability is perhaps even more vital than learning the 'ideal' course of action from a teacher, or attempting to independently extract a latent structure in the environment: two perspectives on machine learning that have arguably received significantly more attention. The generality of RL is, however, certainly impressive: it has been utilized in diverse applications including economics, advertising, optimal control, and robotics, and seems to naturally suit a variety of formulated problems.

Conventionally, RL problems are formally defined in terms of finite Markov Decision Processes (MDP), a classical formalism utilized for sequential decision-making which adequately captures the notions of actions, state, and the consequences of immediate and future rewards. At each time-step t, an agent interacts with its environment through an *action*, a_t, observes the current *state*, s_t, and receives a *reward* signal, r_t.

A successive sequence of alternating states and actions defines a trajectory, usually called a *roll-out*:

$$\tau = (s_0, a_0, s_1, a_1, s_2, a_2, ...) \tag{3.1}$$

An MDP is traditionally represented by a tuple:

$$MDP : \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle, \tag{3.2}$$

where $\mathcal{S}$ and $\mathcal{A}$ are the sets of all states and all actions, respectively.

$\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ (occasionally substituted for $\mathcal{T}$) is a state-transition probability function that governs the transition from one state of an MDP to the next, given the current state s_t and action a_t. Denoted by $\mathcal{P}^a_{ss'}$ or $\mathcal{P}_a(s, s')$, this probability is generally given by:

$$P(s_{t+1} \in S_{t+1} | s_t, a_t) = \int_{S_{t+1}} \mathcal{P}(s_t, a_t, s')ds' \tag{3.3}$$

The definition in Equation (3.3) is general one that respects possibly continuous state spaces, in which the probability of reaching an exact state is technically zero [103]. Instead, we define the probability of reaching a certain encompassing 'region' of the state space, S_{t+1}, hence the integral. In discrete spaces, the value of $\mathcal{P}^a_{ss'}$ clearly reduces to an entry in a probability matrix. Due to this dependence on search space structure, $\mathcal{P}$ can be represented by a simple table, or more sophisticated probabilistic models.

Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ provides the expected immediate reward, r_{t+1}, of transitioning from s_t to s_{t+1} following a_t, sharing the Markovian property of $\mathcal{P}$:

$$r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1}) \tag{3.4}$$

The literature reveals a minor divergence in notation that bears mention here: reward r_t is usually denoted as a function of all three parameters as above, $\mathcal{R}^a_{ss'}$, but the dependence on s_{t+1} is occasionally removed: $\mathcal{R}^a_s$. Both views seem equally plausible and the choice depends on the perceived complexity of the reward 'landscape' and, consequently, the problem at hand.

Unsurprisingly, the reward function has a large impact on learning performance, since it must implicitly encode the goals to be achieved and behavior to be adopted. In fields such as robotics, designers therefore face the task of formulating reward models that express the performance of an agent well enough to promote learning. While binary rewards signifying success or failure are common, *reward shaping* procedures aim to fine-tune reward signals such that closeness to a goal and secondary task considerations are taken into account.

$\mathcal{P}$ together with $\mathcal{R}$, also called the *dynamics* of a MDP, represent a *model* of the environment, and thus play a central role in the distinction between model-free and model-based RL algorithms.

Episodic and Infinite-Horizon Tasks

At this point, it becomes necessary to define the pivotal concepts of episodic and infinite-horizon problems, and discounted returns. Since an agent effectively learns from the aggregation of reward signals over time, a distinction arises between tasks which are accomplished in independent episodes, such as an execution of a hand-over, or persisting tasks, such as optimally managing processes driving a nuclear reactor.

A single roll-out, τ, of an episodic task which ends after T time-steps, having a finite trajectory length, can be naturally evaluated based on the cumulative reward:

$$R(\tau) = r_T + \frac{1}{T}\sum_{t=0}^{T-1} r_t, \tag{3.5}$$

This formulation is derived from [35], whose authors include a term particularly relevant to robot task executions: final reward, r_T, which may reward/punish an effect observable only at the conclusion of a task. Note that the sum in the second term is not always averaged, and can often be encountered without the fraction, depending on the problem.

In contrast, an infinite-horizon task does not necessarily terminate, leading to the notion of a factor that discounts rewards farther away in time, $\gamma \in [0, 1)$. Adjusting our definition of accumulated reward, we arrive at:

$$G_t(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{3.6}$$

which we now define as the *return* of a roll-out: the sum of cumulative, discounted future reward from time-step t onwards, where we opt to use G_t to elucidate this distinction (as in Sutton et. al.'s [103]).

Intuitively, smaller values of *discount factor* γ favour myopic evaluations of the return, while values approaching 1 are more 'far-sighted', considering the consequences of actions taken farther in the future. Primarily, this is mathematically convenient, since it allows working with an otherwise infinite sum, thus avoiding unbounded accumulation of reward signals, some of which may lose their significance over time, in the absence of a definite termination condition. Moreover, it captures the instinctive preference of shorter-term rewards over delayed ones, while not forgoing the relevance of the latter. It also represents uncertainty in received rewards, which obviously rises further along the horizon.

Policies

In the most general sense, a *policy* sufficiently encapsulates the behavior of the learning agent, and is the central element of RL. A policy, π, maps observed states to probabilities of selecting the possible actions (or, simply, to actions), and thus defines a probability distribution over actions $a \in \mathcal{A}$, given states $s \in \mathcal{S}$:

$$\pi(a_t|s_t) = P(a_t|s_t), \tag{3.7}$$

in which case actions are sampled from a *stochastic* policy: $a_t \sim \pi(\cdot|s_t)$. This is the case, in general, as opposed to a *deterministic* policy: $a_t = \pi(s_t)$. Occasionally, stochastic policies are represented instead as a joint distribution, $\pi(a_t, s_t)$, instead of the usual conditional, signifying the probability of being in state s_t and taking action a_t.

A policy can take various forms, depending on the nature and complexity of the problem, and the sophistication of the desired behavior, including simple functions, lookup tables, controllers, probabilistic models, stochastic processes, neural networks, and motor primitives.

The ultimate goal in RL is experiential learning of the *optimal policy*, $\pi^*(a|s)$, which maximizes

the average expected return, roughly formalized as

$$J_\pi(\tau) = \int_s \int_a \mu^\pi(s)\pi(a|s)\mathcal{R}(s,a)dsda, \tag{3.8}$$

where J_π signifies a performance measure of policy π, $\mu^\pi(s)$ denotes a stationary state distribution resulting from following π, subject to some transition dynamics, $\mathcal{P}$. Here, we remove explicit dependence of the reward function on the next state s_{t+1}, for notational convenience.

Note on Notation in Robotics In the field of robotics, low-level robot actions are usually in the form of joint commands, while the states usually encapsulate robot joint coordinates, and occasionally, aspects of the environment or objects relevant to the task. Consequently, the literature on RL in robotics often utilizes a different notation for policies:

$$\pi(u_t|x_t) = P(u_t|x_t), \tag{3.9}$$

In this report, we place no emphasis on either, but adopt this convention in our own work. In cases of potential ambiguity within the text, the constituents of the used notation and what they stand for are plainly clarified.

Value Functions

Since J_π must be estimated from each visited state, as the 'cost-to-go', following a given policy, it is convenient to introduce two *value functions*. *State-value function* $V^\pi(s)$ (alternatively: $v_\pi(s)$) represents the expected return when an agent starts at s_t, and follows π thereafter, and is analogous to the 'value' of being in that state:

$$V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] \tag{3.10}$$

This value provides a far-sighted measure of a state's desirability, considering the states that will most likely follow, and the rewards we expect to consequently receive. In essence, this facilitates targeting actions that increase the likelihood of observing states with bigger value, and thus indirectly ensures our future rewards are maximized.

Action-value function $Q^\pi(s,a)$, or $q_\pi(s,a)$, simply extends the state-value function by explicitly depending on a particular action as well as a state, to estimate expected return:

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] \tag{3.11}$$

In traditional, *value-based* RL, $\pi^*(a|s)$ is derived by first determining the optimal value functions, $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, particularly the latter, which basically show the maximum expected return that can be possibly achieved by **any** policy, for a state or a state-action pair, respectively. Given the theoretically proven optimality of a policy whose value functions are optimal [103], it follows that a *greedy* policy: one that samples actions that maximize these values, must be an optimal

one. This gives rise to a systematic method for finding optimal policies and thus 'solving' a MDP, which will be discussed further in the next sub-section.

Exploration and Exploitation

The sequential decision-making aspect of RL, in contrast to other machine learning paradigms, introduces the challenge of constantly choosing between either making the best decision, given the current information, or gathering more information, through sub-optimal decisions: an *exploration-exploitation trade-off*. Paradoxically, we would like an agent to exploit acquired experience to maximise reward, but as a consequence of the interactive learning process, we realize it must also explore to access uncharted and potentially more promising regions of its search space. Intriguingly, solely pursuing either of the extremes guarantees an agent's failure in a task framed as an RL problem.

In this dilemma, striking a balance hinges on a realization that the best long-term strategy, and thus policy, may require occasional sacrifices, and that information-gathering pursuits are as vital to achieve that as capitalising on current knowledge. This matter is taken into consideration in policy representations, such as inherently stochastic probabilistic models, or by applying various strategies such as naive exploration strategies, probability matching (such as is used in *Thompson sampling*), and the so-called information state search, which targets areas of high uncertainty (in information-theoretic terms, it attempts to maximise entropy). Another oft-used technique is learning off-policy.

On-policy and Off-policy Learning

An important dichotomy in RL is that of an agent learning *on-policy* or *off-policy*. The simpler and more rudimentary of the two, on-policy learning, involves a process of improving or optimizing the same policy from which actions are sampled, thus trying to learn optimal behavior while acquiring experience through the behavior learned up until that point. Off-policy learning, on the other hand, has the same objective, but samples actions from a different policy and uses the acquired information to update the policy to be learned. The one from which data is generated is termed a *behavior policy*, while the one being learned is the *target policy*.

The game of Chess, a popular test-bed in RL, provides a useful analogy. An inexperienced player starts playing with little insight on the value of each board configuration or of their moves, and thus follows an inferior policy. Over time, they suffer consequences of bad moves and benefit from good ones, incorporating this information in future decisions from which they acquire even more experience, and re-iterate: they learn on-policy. Conversely, a (particularly patient) player can decide to play many games while choosing moves completely randomly, while observing and assimilating the results of their decisions in a similar way, opting to encounter myriad, often futile, scenarios before putting the shaped strategies to practice. They thus acquire data to learn a good policy 'off' that policy; in this case, from some other random policy.

Although marginally non-intuitive, off-policy learning addresses a dilemma on-policy strategies face: trying to simultaneously follow and learn the optimal behavior, sampling actions from this policy, while it is known that exploring, i.e. behaving sub-optimally, is a requisite for eventually finding the

optimal actions. Lack of exploration is usually mitigated by learning behavior under a sub-optimal policy that explores to some extent, instead of pursuing the optimal. An adjacent solution, sometimes called *naive exploration*, is ϵ-*greedy* policies, which either choose the highest-valued action, or pick a random action with a probability, ϵ, at each time-step.

Tackling the exploration-exploitation trade-off by splitting the two sides of the coin into separate policies is therefore often a practical alternative, since one can conveniently engineer the desired rate and form of exploration, while not degrading the target policy. The concept of *importance sampling* is central to off-policy learning, and involves estimating expected values of a distribution, using samples from another. Despite the added complexity, off-policy learning is considered more general and effective. As would be expected, however, it suffers from slower convergence, mainly due to greater variances in the data which is generated from a different, stationary distribution.

Model-free and Model-based RL

So far, we have made no mention of the presence or absence of a *model*, a key element in RL systems which defines whether we can infer how the environment behaves in response to our agent's action. In the MDP formulation, this is synonymous with the state-transition and reward functions which, when available, can be exploited to obtain the optimal value functions and thus policy, since we would then know exactly how probable it is to transition from a state to another following an action, and what reward the agent would receive. This allows us to look ahead when planning, an essential element of *dynamic programming* and *temporal difference* algorithms, for example. Needless to say, we seldom have this information and, therefore, any MDP at all, particularly in difficult problems like those in robotics, but the groundwork we have laid using MDP formulations still equips us with the tools to tackle the model-free case.

Model-free RL involves a strictly trial-and-error approach, devoid of any 'planning', that attempts to learn an optimal policy purely from active experience, given no information about environment dynamics. In other words, the value functions and/or policy are optimized by simply trying actions and observing immediate rewards, avoiding any modelling. With no indication on values of actions or expected return, it is apparent that a large number of trials would be required to arrive at acceptable results. This is even more troublesome on systems such as robots, for which safety and cost become significant concerns.

As the name implies, model-based algorithms make use of a model, if available, but more interesting variants attempt to jointly learn a model in addition to the value functions and/or policy. Whether learned or provided, a model of transition dynamics enables predicting rewards and optimal actions, providing a structure to an agent's search and thus better indication on directions to pursue. As a result, learning can be performed 'off-line', which is much more *sample-efficient*: the most prominent advantage of using models. However, these algorithms have a notorious tendency to exploit model inaccuracies, artificially minimising costs to produce policies that may not translate well to the real system. Moreover, models may be too difficult and costly to construct, if at all feasible. Independence on prior knowledge and easier implementability are thus marks in favour of model-free algorithms, despite the clear advantages provided by the alternative, in terms of performance.

Inverse Reinforcement Learning (IRL)

As previously mentioned, reward functions/models substantially influence RL and performance thereof, prompting intricate designs through reward shaping. An alternative to hand-crafting models, which may be sub-optimal due to the inherent biases they introduce, is IRL, which involves extracting a reward model from expert demonstrations. LfD (discussed in section 2.1.3) is usually done in a supervised fashion, where supposedly optimal behavior is learned directly. Alternatively, an agent can 'indirectly' learn the implied policy by extracting reward signals when observing a demonstrator, then utilizing these to reinforce its own learning [103]. This is equivalent to learning a task description that implicitly captures the demonstrator's goal(s).

IRL essentially formalises the learning of a MDP's reward function, usually assuming state-transition dynamics are available and given observed demonstration trajectories in the form of state-action pairs [75]:

$$\tau = \{(s_0, a_0), (s_1, a_1), (s_2, a_2), ...\} \tag{3.12}$$

Algorithms then produce a reward model that maximizes the likelihood that the observations came from the reward-augmented MDP. To that end, trajectory data is used to infer a structure for the reward which the expert had attempted to maximise in achieving their (possibly unknown) goals. Matching the demonstrator in this form of LfD thus emphasizes inference of some transferable description of the task to be performed.

A fairly common approach to IRL is a non-parametric Bayesian segmentation of demonstrated trajectories, by extracting and segmenting their underlying reward functions, which is primarily used to discover multiple 'skills' from unstructured demonstrations [93].

One of biggest difficulties with IRL is the fact that it is an ill-posed problem: given state-actions pairs of τ, constructing a function that assigns a unique constant to each makes them all trivially 'optimal', which complicates the task of extracting a meaningful policy. Certain restrictions on model structure, such as linearity, and other constraining assumptions have enabled inferring proper reward models. This, of course, leads to a host of other difficulties, and is among the reasons IRL remains a difficult problem.

3.1.1 Value-based Reinforcement Learning

As previously hinted at, under the assumption that we know a problem's finite MDP, we can find the optimal value functions which are used to reconstruct a policy that optimally solves the problem. In RL, searching for this policy can be framed as an optimization problem which can be solved in its primal formulation, by searching in the space of possible policies, or by solving this dual formulation, which gives rise to *value-based* RL algorithms, occasionally called *action-value methods*. This indirect solution to the problem is motivated by the ability to leverage the *Bellman Principle of Optimality*, which facilitates an efficient, recursive learning procedure, subject to some assumptions.

Bellman Optimality Equations

Bellman's principle states that: "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [13]. The implied strategy is simplifying a complex or intractable problem by breaking it into sub-problems and solving these constituent parts recursively. It addresses the complexity of traversing the whole search space of states and actions in order to determine the expected reward and, subsequently, the optimal state-value or action-value, for each visited state.

First, the *Bellman expectation equations* decompose the value of a state/state-action pair into the immediate reward and the discounted value of the succeeding state:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s], \tag{3.13}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})|s_t = s, a_t = a], \tag{3.14}$$

which are simply derived from equations (3.10) and (3.11), and the fact that (from equation (3.6)) $G_t = r_1 + \sum_{t=2}^{\infty} \gamma^t r_t$, since the reward observed at the current state is not discounted. The rationale, here, is that the value of a state is the immediate reward received when entering that state plus an estimate of the total reward that would be received by acting optimally from there onwards (until episode termination, or hitting the horizon limit). This concept of *bootstrapping* is the focus of Bellman's methods.

Subsequently, the *Bellman optimality equations* constitute consistency conditions that are satisfied only by the optimal value functions, which are then given by:

$$V^{*}(s) = \max_{a} \int_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'}[\mathcal{R}^{a}_{s} + \gamma V^{*}(s')] \tag{3.15}$$

$$Q^{*}(s,a) = \int_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'}[\mathcal{R}^{a}_{s} + \gamma \max_{a} Q^{*}(s', a')] \tag{3.16}$$

Note that the integral can be substituted with a summation for discrete state spaces, and that the expectation has been made explicit by including the probabilities $\mathcal{P}^{a}_{ss'}$, and the integration over all states. r_{t+1} has also been replaced by $\mathcal{R}^{a}_{s}$ to expose the dependence on action and state, keeping in mind that the $t+1$ subscript of the former is due to the fact that rewards are, by convention, received at the **next** time-step.

An optimal policy would always choose actions that maximise Q values, according to $Q^{*}(s,a)$, at each time-step: $\pi^{*}(a|s) : \forall t, a_t = \arg\max_{a \in \mathcal{A}} Q^{*}(s,a)$. The implications of this are that any MDP has a deterministic optimal policy (in that we can predict what action it would sample), and that simply knowing $Q^{*}(s,a)$ leads us to the optimal policy, one that is greedy with respect to $V^{*}(s)$. This last assertion directly follows from:

$$V^{*}(s) = \max_{a} Q^{*}(s,a) \tag{3.17}$$

The Bellman equations essentially provide an easier analytical solution to the problem of optimizing J_{π}, which warrants using the value-based dual of the policy search problem. It is interesting to note that

this powerful tool is enabled by a simple question answered by the value functions at each time-step: if I am in this state and take this action, assuming I take the optimal action at every subsequent time-step, what return can I expect, i.e. what is this state's value?

Dynamic Programming (DP)

The recursive procedures contained in equations (3.13), (3.14), (3.15) and (3.16) form the basis of the well-known and general approach of *dynamic programming* (DP), in which this property of the Bellman equations in conjunction with the cache-and-reuse property of value functions enables iteratively evaluating and improving a policy, our central objective in RL.

The two stages of DP are:

1. *Policy evaluation*: using equations (3.13) and (3.14), a policy is evaluated by gradually converging on a stable estimate of the values, $V^\pi(s)$, of all states $s \in \mathcal{S}$. The equations are applied iteratively, synchronously updating all state-value estimates by finding the old values of all successor states, s', and expected immediate rewards, for all possible one-step transitions. The update rule is then (similar to eq. 3.13):
$$V(s_t) \leftarrow \mathbb{E}_\pi[r_{t+1} + \gamma V(s_{t+1})], \tag{3.18}$$
This is sometimes referred to as the prediction problem.

2. *Policy improvement*: using equations (3.15) and (3.16), the optimal value function, $V^*(s)$, for all $s \in \mathcal{S}$, is calculated and a new policy is synthesized by acting greedily with respect to its values. Due to the monotonicity, the result is a policy that is guaranteed to be an improvement, except if it is π^*.

Combining the two leads to the general *policy iteration* algorithm, which iteratively evaluates and improves the policy, converging to the optimal (as depicted in [103]):

$$\pi^0 \xrightarrow{E} V^{\pi^0} \xrightarrow{I} \pi^1 \xrightarrow{E} V^{\pi^1} \xrightarrow{I} \pi^2 \xrightarrow{E} ... \xrightarrow{I} \pi^* \xrightarrow{E} V^* \tag{3.19}$$

The bootstrapping performed when updating state-value estimates enables the algorithm to (relatively) tractably refine values of all parallel states and store them for the complete state space.

A slightly different variant, called *value iteration*, follows the same procedure, except that the policy evaluation step is truncated: instead of waiting until it converges to an estimate of $V^\pi(s)$, it is stopped after a single sweep (or 'backup') in the expectation equations, and the policy is improved immediately. This removes the iteration loop of the evaluation step, which is part of the main loop, thus theoretically speeding up convergence to π^*. *Generalised policy iteration* encapsulates a class of methods in which any applicable algorithm can be used for the evaluation and improvement steps, respectively.

The main drawbacks of DP are an assumption of a perfect model, since it wholly relies on a complete MDP being provided, and the computational expenses due to the exhaustive recursive procedure.

Monte Carlo (MC) Methods

Monte Carlo (MC) methods are an alternative, model-free class of value-based RL algorithms that learns value functions and the optimal policy strictly from experience, by sampling episodes. Particularly effective when MDPs are unknown, they substitute the bootstrapping technique used in DP to estimate the expected return from a state (i.e. value) by observing trajectory samples: instead of the expected value, MC methods use a much simpler estimate based on the empirical mean of returns for each state, as observed from experience:

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - \gamma V(s_t)), \tag{3.20}$$

wherein value $V(s_t)$ is updated toward actual, observed return G_t every time it is visited, $\alpha = \frac{1}{N(s_t)}$, and $N(s_t)$ keeps a count of these occurrences.

Since MC algorithms do not bootstrap, returns cannot be estimated except by reaching terminal states, at which point the exact return of all visited states is known, for that episode. The value of each s is then the incremental mean of the returns observed across episodes, an unbiased approximation that tends to the actual value of s. Otherwise, the same policy iteration procedure used in DP algorithms applies to MC algorithms as well.

The greatest appeal of MC RL lies in learning purely from experience, despite no knowledge of the environment, which can still lead to finding optimal policies. It also enables sampling from simulations, which only have to provide sample transitions, as opposed to complete models. Another added advantage is removal of the dependence on the Markov assumption: since MC algorithms do not bootstrap, they are equally effective in non-Markov environments. Crucially, however, these algorithms must observe complete trajectories before evaluating and improving a policy, since returns are unknown until an episode ends. Consequently, MC is only applicable to episodic scenarios, since termination is a vital component here.

Temporal Difference (TD) Learning

The third and most widely used class of value-based methods are temporal difference (TD) algorithms, which are similarly model-free and learn from actual experience. Their novelty lies in the combination of bootstrapping (from DP) and sampling (from MC), such that this experience is gained from incomplete episodes. The policy evaluation procedure then involves updating state-value estimates towards a one-step estimate of the return, instead of the actual values, as in MC:

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)), \tag{3.21}$$

where, instead of G_t in MC, values are updated towards the estimated return $(r_{t+1} + \gamma V(s_t))$, called the *TD target*. In addition, $\delta_t = (r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$ is called the *TD error*, with the subtraction of the latter two terms giving rise to the method's name, since it signifies the temporal difference in predictions, which is used to compute this 'error'.

In essence, the sampling of TD algorithms avoids the assumptions of DP while relaxing the restrictions of MC, consolidating the two in what is usually the optimal approach. In particular, TD

methods can be more readily applied for on-line learning, since they wait until the next time-step to obtain an estimate of the return, instead of waiting until an episode ends. The ability to learn from incomplete sequences also extends their applicability to non-terminating environments.

It is worth noting that TD algorithms are known to be sensitive to initial state-values. In addition, the subsequent estimates, while they suffer from less variance than those of MC algorithms, are now biased since the TD target does not rely on actual returns, but estimations of the current values of states: a known effect of bootstrapping.

Nevertheless, TD algorithms are the most frequently applied class of critic-only algorithms, and form the basis of the important Q-learning and SARSA algorithms, among others. Most of these apply the n-step variant of TD learning, called *TD(λ)*, in which the steps of look-ahead are varied according to requirements by setting a value for λ other than the default 0. This forms a spectrum of possible algorithms that exist in a continuum between DP and MC methods.

Value Function Approximation

The reliance of value-based RL algorithms on caching and re-using state-value estimates is occasionally an obvious liability, particularly for complex problems, such as in robotics applications, with vast state spaces. Under an assumption that we work with discretized state spaces, this leads to problems with very large MDPs, containing a number of states and actions that makes storing $V(s)$ and/or $Q(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ in lookup tables largely infeasible. An alternative that is borrowed from supervised machine learning approaches is to approximate the two value functions, which are then represented in a parameterized functional form:

$$\hat{V}(s, \mathbf{w}) \approx V^{\pi}(s) \tag{3.22}$$

Using function approximation, the value function under policy π is approximated by a function that depends on a weight vector. In this manner, it is easy to incorporate value information from observed states through a regression procedure, for example, and subsequently generalise to unvisited states, systematically estimating their values. Conveniently, the established classes of algorithms, MC and TD, can be employed to learn the parameters w that best approximate the value function landscape. The result is a much more memory-efficient RL procedure, when computation of the value functions is a necessary requisite to finding the optimal policy.

As one can imagine, a range of functions and models can be used to approximate the value function, the simplest one being a combination of features that characterize a state (usually denoted by x, in agreement with classical machine learning notation), which is linear in the weight vector w. Other likely candidates include neural networks, decision trees, and Gaussian Processes, among others. The concept of *deep reinforcement learning* actually originates from the use of deep neural networks as approximators for value functions.

Naturally, it is conceivable that the requirement of accommodating complex and memory-intensive value functions can be eliminated by not using value functions to derive the optimal policy at all, which is what the algorithms in section 3.1.2 apply.

Notable Algorithms

The following is a list of notable and pioneering value-based reinforcement learning algorithms, most of which have been formalized by Sutton et. al. in [103].

- Q-learning:
  A basic off-policy, model-free, value-based algorithm that essentially maintains a table of Q values corresponding to all state-action pairs, and performs an exhaustive DP policy iteration procedure (as previously described). In essence, the Bellman optimality equations are leveraged to determine the optimal values, Q^*, from which a greedy policy that maximizes these is derived, π^*.

- SARSA:
  An on-policy algorithm that resembles Q-learning but computes Q values based on the current 'learned' policy, instead of a greedy, Q-maximizing policy (SARSA is an acronym for State-Action-Reward-State-Action).

- TD(λ):
  The most prominent temporal difference learning algorithm, TD(λ) estimates value functions by sampling episodes and bootstrapping from current estimates till the end of the horizon, essentially constituting a hybrid of MC and DP algorithms. The λ parameter controls the degree of look-ahead in the algorithm's bootstrapping procedure.

- Deep Q-Networks (DQN) [78]:
  A variant of Q-learning that utilizes neural networks to approximate the Q-value function, as opposed to look-up tables, in order to scale the algorithm up to higher-dimensional and more expansive search spaces. The algorithm was among the first 'deep reinforcement learning' algorithms, and achieved various breakthroughs in benchmark problems.

Despite the satisfying solutions provided by the Bellman equations, and the capabilities of TD algorithms, value-based RL approaches suffer from significant drawbacks. Firstly, the value functions must be computed to derive the policies that they imply, an extra step in the process which may also implicitly limit the structure of the reconstructed policy. Additionally, this policy may degrade due to biases introduced through bootstrapping, as well as discontinuities in the value function, which introduce errors that can propagate through recursions. It is also inefficient and difficult to apply in high-dimensional state and action spaces, and virtually inapplicable to continuous ones. The aggregation of these issues makes policy search a more favourable strategy, particularly for problems in robotics.

3.1.2 Policy-based Reinforcement Learning (Policy Search)

We now consider the primal formulation of the constrained optimization problem, instead of the aforementioned dual, for which J_π in Equation (3.8) is maximized: searching for the optimal policy directly

in the space of policies, instead of computing value functions and subsequently deriving them. This is done by learning a parametrized policy, $\pi(a|s,\theta)$ (or π_θ) that maximises the performance measure, or objective function, J_θ. (Note the conscious change from J_π to J_θ symbolizing the more explicit reliance on the policy parameters, which are the target of optimization in PS.) The primary motivation, here, is to better scale RL to high-dimensional, continuous spaces, which is achieved by avoiding value functions and using parameterized policies that restrict the space of policies for more tractability.

The most prevalent form of policy search (PS) algorithms are policy gradient methods, which learn optimal parameters θ by performing stochastic gradient ascent (SGA) on J_θ, such that a likely parameter update rule is:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J_{\theta_t} \tag{3.23}$$

This is known to achieve learning that exceeds action-value methods in speed, and works with a plethora of policy formulations. Obviously, an important requirement is that $\pi(a|s,\theta)$ be continuously differentiable with respect to θ, since the underlying *policy gradient theorem* implies updating the parameters in the direction of an estimated gradient of the objective function, with respect to these parameters, at each time-step.

PS algorithms that do not necessarily use hill climbing strategies employ Expectation-Maximization-based and information-theoretic-based policy updates, each of which are discussed in further sub-sections, as well as less common evolutionary and Bayesian optimization strategies.

Using PS, we can learn well-formulated stochastic policies with various advantages, such as implicitly encoding exploration, and allowing to specify its desired properties. A simple example is a Gaussian policy:

$$\pi_\theta(a|s) = \mathcal{N}(a|\mu_a(s), \Sigma_a) \tag{3.24}$$

where parameters $\theta = \{\mu, \Sigma\}$, and covariance matrix Σ would be chosen to achieve the required exploration rate.

A clear advantage lies in the smaller set of parameters required to learn a policy directly, than learning an optimal value function. It also follows that PS is naturally less prone to the adverse effects of the *curse of dimensionality*, better handling high-dimensional search spaces. The parameterization also significantly mitigates the difficulties normally faced with continuous spaces, whose discretizations may be similarly unmanageable.

By virtue of the variety of policy representations one may choose, ranging from linear controllers and radial basis function networks, to time-dependent dynamic movement primitives ($\pi_\theta(a|s,t)$), the potential to integrate expert knowledge into the policy, both to bootstrap learning and dictate the learning process, is now realized. A typical example is our use of demonstrations to initalize a policy for a hand-over task from a demonstrator using LfD (refer to section 2.1.3). This capability, which does not translate to value-based formulations, makes PS much more suitable for complex real-world problems, particularly in robotics.

After having introduced PS algorithms, we briefly look into the distinction having a model or not has on their nature and properties, each of the three common policy update strategies (and thus, classes

of PS algorithms), and the contextual extension to PS.

Model-free and Model-based Policy Search

The distinction between model-based and model-free algorithms in RL, introduced at the beginning of this section, bears re-iterating for the case of PS, since it has vital implications, not generally shared by action-value methods, that are particularly significant for robotics applications. In general, the basic definitions of each remains as described before.

Model-free PS algorithms, when applied in robotics, tend to involve sampling real robot trajectories when trying to execute some task, then using the returns obtained from these to directly alter and optimize the current policy. This is termed stochastic trajectory generation, such that the state transitions (dictated by the underlying and unknown $\mathcal{P}$) are sampled from the real robot, leaving only the reward signal as a requirement for policy search. In this context, it is useful to represent the distribution over (observed) trajectories (τ) [35]:

$$P_\theta(\tau) = P(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t, t), \tag{3.25}$$

leading to the slight reformulation of Equation (3.8) to fit the PS case:

$$J_\theta(\tau) = \int_\tau P_\theta(\tau)\mathcal{R}(\tau)d\tau, \tag{3.26}$$

Note the use of a time-dependent stochastic policy, which is usually the case in a robot task policy. Maximising Equation (3.26) is the objective, as before, and can be interpreted as choosing θ such that the expected reward that can be obtained over the distribution of possible trajectories is maximised. As before, running this directly on a real robot would require a vast number of interactions, which may be both costly and dangerous.

The model-based version of PS seeks to avoid this inefficiency by constructing *forward models* of the robot's and environment's dynamics, from the observed trajectories, and using these to internally simulate roll-outs, whose results are then used to update the policy, before transferring it back to the physical system. In order to improve the learned models over time, the policy is executed on the robot to gather more data with which the refine the model, and the loop continues. These models provide a means to obtain a large number of sample trajectories without the drawbacks of physical trials, and the long system interaction times.

A significant challenge with model-based PS is mitigating the effects of model errors, which may strongly bias the learning process, and lead to sub-optimal policies that do not transfer well to the system. This is usually tackled by learning non-parametric probabilistic forward models, such as Gaussian Processes (like in PILCO and GPREPS) and Locally-weighted Bayesian Regression (LWBR), which both model uncertainties in predictions and increase robustness to model inaccuracies. Capturing faithful models is, however, also difficult without large amounts of input data, especially for complex robot tasks, for example, which may then nullify the very purpose of these models.

The verdict on the matter is that model-free approaches are easier to implement since they need no

forward models to learn and thus avoid the inherent biases, but model-based approaches are progressively taking over as the prime choice for faster and more data-efficient learning, particularly in robotics. Nevertheless, the issue of determining and optimizing good forward models remains a challenge.

Policy Gradient-based Policy Search

Policy gradient (PG) methods for PS, as described before, optimize policy parameters by performing stochastic gradient ascent (SGA) on the objective function J_θ, with the update rule shown in Equation (3.23) guided by the gradient estimate given by (from Equation (3.26)):

$$\nabla_\theta J_\theta(\tau) = \int_\tau \nabla_\theta P_\theta(\tau)\mathcal{R}(\tau)d\tau, \tag{3.27}$$

Estimating this gradient, which amounts to estimating $\nabla_\theta P_\theta(\tau)$, can be done in one of several ways. Finite difference methods rely on random perturbations to θ for which the change in return is computed, and which then lead to a gradient that can be computed using a Taylor-expansion of J_θ [35]. A class of algorithms use likelihood-ratio policy gradients, which are obtained by applying the log-likelihood trick to $\nabla_\theta P_\theta(\tau)$ such that a resulting expectation on $P_\theta(\tau)$ can be approximated using a sum over trajectories, leading to an unbiased estimate of the gradient. This is applied in the famous REINFORCE algorithm. Finally, natural gradients were introduced to provide an estimate that addresses the drawbacks of the SGA approach, mainly the likely presence of local minima, which harm PG algorithms' performance. The aim is to incorporate knowledge about the landscape's curvature into the gradient, and an important aspect is limiting the step-width between two subsequent trajectory distributions, $P_\theta(\tau)$. Natural gradients are the main component of Natural Actor-Critic algorithms.

PG algorithms are notable for their favourable convergence properties, thanks to the application of gradient ascent, but they remain dependent on the calculation of the gradient in some way, and policy gradients are known to be problematic due to high variance, especially in robot platforms. They are also inevitably governed by the values of a learning rate.

EM-based Policy Search

Gradient-free methods are generally less demanding than their opposite, since no computation of a gradient is required, placing no differentiability constraints on the policy model. One such class of methods relies on probabilistic inference for inferring the policy, instead of estimating its distribution from the given data. It is based on Expectation-Maximization (EM), which enables finding maximum likelihood solutions for probabilistic models involving a hidden/latent variable. In a case such as PS, this has been applied to infer what trajectories in the space of trajectories obtain high rewards, leading to what would be inferred as the optimal policy (or policies).

EM-based PS algorithms model the problem of RL as a maximum likelihood problem, taking trajectories τ as the latent variables in a model that expresses reward conditioned on these variables, $P(R|\tau)$, and aiming to find parameter vector θ such that the probability of high reward is maximised or

equivalently, the MLE solution for the log-marginal likelihood:

$$logP_\theta(R) = \int_\tau P_R(\tau)P_\theta(\tau)d\tau, \tag{3.28}$$

The elaborate details of the actual implementation of the EM algorithm are omitted here, but the interested reader is directed to a succinct overview provided in [35].

The EM-based formulation of PS has lead to various influential algorithms, including Policy learning by Weighting Exploration with Returns (PoWER), Cost-Regularized Kernel Regression (CrKR): one of the first multi-contextual PS algorithms, and Reward-Weighted Regression (RWR).

Information-theoretic Policy Search

Information-theoretic PS algorithms similarly do not rely on policy gradients, instead guiding policy updates by bounding information loss between subsequent updates, while maximizing expected reward, in a constrained optimization problem. These algorithms focus on optimizing the policy while staying close to the observed data, by minimizing the KL-divergence, also called the relative entropy, between the previous trajectory distribution, q, and the distribution generated by the new policy, p. This effectively bounds steps between subsequent policy updates, reducing loss of previously gathered data, and providing an implicit measure for controlling the exploration-exploitation trade-off.

A simple form of the constrained optimization problem employed by information-theoretic algorithms is:

$$\max_\pi \int \pi_\theta \mathcal{R}_\theta d\theta \tag{3.29}$$

$$\text{s.t.} \int \pi_\theta log\frac{\pi_\theta}{q_\theta} d\theta \leq \epsilon \tag{3.30}$$

$$\int \pi_\theta d\theta = 1 \tag{3.31}$$

The objective of this problem is to maximize expected reward while bounding relative entropy, which addresses premature convergence and stability issues associated with EM-based algorithms, while avoiding learning rates of PG methods ([35]). This is employed by algorithms such as REPS and its variants, which optimize policies by searching in the distribution of trajectories, prioritizing minimal relative entropy in policy updates. Among the advantages of this method is avoiding unstructured exploration, which may be undesirable in safety-critical systems, and maintaining the main characteristics of an initial policy, which is a favourable property for RL with pre-structured policies obtained from expert demonstrations.

Contextual Policy Search

This part briefly explains the concepts underlying a class of context-dependent policy search algorithms.

Contextual policy search (CPS) algorithms enable learning behavioural policies whose output

is conditioned on context variable(s), c, such that an agent's actions generalize to different situations, operating conditions, and/or task variations, without having to learn separate policies. This makes them particularly suitable for the problem of constructing generalizable motor skills, which usually involve episodic task executions.

CPS maintains a search distribution, denoted by $\pi(\omega|c)$, over some policy's parameters ω conditioned on context c, and aims to find the distribution that maximizes objective performance measure $R_{c\omega}$: the expected return of a task episode in context c, while using parameters ω:

$$R_{c\omega} = \mathbb{E}_\tau[r(\tau, c)|c, \omega] = \int_\tau p(\tau|c, \omega)\mathcal{R}(\tau, c)d\tau, \tag{3.32}$$

$p(\tau|c, \omega)$ here refers to a distribution over all possible trajectories, τ, while $\mathcal{R}(\tau, c)$ is an underlying reward function.

Introducing a distribution of over contexts, $\mu(c)$, the objective of CPS is then to maximize:

$$J_\pi = \int_c \int_\omega \mu(c)\pi(\omega|c)R_{c\omega}dsd\omega \tag{3.33}$$

by adequately choosing the parameters ω of the execution policy.

These algorithms execute roll-outs by observing context c, sampling parameters ω, and following the execution policy, subsequently receiving reward $R_{c\omega}$. As a result, data samples $\mathcal{D} = \{c^{[i]}, \omega^{[i]}, R_{c\omega}^{[i]}\}_{i=1,\dots,N}$ from the old search distribution, usually denoted by $q(\omega|c)$, are used to compute a new search distribution, $\pi(\omega|c)$, that maximizes J_π. As $\pi(\omega|c)$ is often a linear-Gaussian model, this is achieved by computing a weight for each sample i (according to $R_{c\omega}^{[i]}$), and performing a weighted maximum likelihood estimation (WMLE) update of the Gaussian distribution.

Traditionally, the search distribution is termed the upper-level policy while the policy which is parameterized by the former and controls actual task execution is the lower-level policy. This creates a hierarchical policy structure that effectively generalizes the lower-level policy to multiple contexts, or solution modes, and facilitates the intended contextual adaptivity.

Examples of algorithms that employ CPS include Contextual Relative Entropy Policy Search (C-REPS), which we employ to address the problem presented in this project, and Contextual Model-based Relative Entropy Stochastic Search (C-MORE). Both algorithms rely on information-theoretic policy update strategies.

Notable Algorithms

The following is a list of notable policy search algorithms. An extensive and recommendable survey on policy search algorithms is provided by Deisenroth et. al. in [35].

- REINFORCE [117]:
  One of the first PG algorithms, REINFORCE relies on Monte-Carlo roll-outs to estimate episode returns and calculates policy gradients using a likelihood-ratio trick. It is also among the first approaches that explore in the action space by adding normally-distributed noise directly to actions.

- Natural Actor-Critic (NAC) [87][88]:
  A PG algorithm that applies a temporal difference learning approach and is distinguished by the use of natural gradients, which incorporate knowledge about search space curvature, and thus lead to faster convergence in more challenging landscapes. Since this gradient is independent of policy parametrizations, NAC is better able to handle arbitrary policy formulations. Exploration is performed by searching directly in the parameter space.

- Cost-Regularized Kernel Regression (CrKR) [61]:
  A kernelized version of the EM-based Reward-Weighted Regression (RWR) algorithm, and among the first designed to solve contextual problems, by learning a generalizing, upper-level policy. Employing a GP policy formulation, the algorithm's update strategy corresponds to GPR. It applies an uncorrelated exploration strategy, which is generally undesirable.

- Policy learning by Weighting Exploration with Returns (PoWER) [60]:
  The most prominent EM-based PS algorithm, PoWER searches in the parameter space locally, applying a structured exploration strategy, and reuses roll-outs through importance sampling. The update strategy closely resembles, but greatly improves upon, that of RWR. It is particularly suitable for motor primitive learning in robotics, outperforming most applicable algorithms ([59]).

- Relative Entropy Policy Search (REPS) [89]:
  An information-theoretic PS algorithm that searches for the optimal policy in a constrained optimization problem, which prioritizes bounding information loss (relative entropy) between subsequent policy updates, such that unstructured and aggressive exploration are avoided. The algorithm utilizes the generally recommended policy update strategy ([35]), and spawned various useful extensions such as C-REPS [67], HiREPS [29], GPREPS [66], and NP-REPS [111].

- Policy Improvement by Path Integrals (PI^2) [107]:
  Derived from principles of stochastic optimal control and the quantum mechanics concept of path integrals, PI^2 was designed for directly optimizing dynamical system policies, such as DMPs, by incorporating exploration noise directly into their equations. Its shares some similarities to PoWER and REPS, such as the computation of sample weights for policy updates.

3.2 Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) are a convenient motion representation, in the form of non-linear dynamical systems, that is often used in robotics to encode trajectories for rhythmic or discrete, goal-directed motions. They were introduced by Ijspeert et. al. to facilitate capturing elementary motions, or motor primitives, in a compact form that allows efficient reproduction, learning and generalization ([50]), and have since been extended and modified to integrate capabilities such as obstacle avoidance ([45]) and improvement through RL. DMPs can also be defined as control policies governed by a set of non-linear differential equations with stable attractor dynamics, that possess particularly useful properties for robot trajectory manipulations. We provide here a brief discussion on trajectory representation using

DMPs, with a focus on goal-directed motions. For a more comprehensive overview, refer to Ijspeert et. al.'s seminal work in [52].

Using the DMP formulation, the evolution of a variable y, representing a terminating motion that starts at y_0 and ends at goal state g, is represented as a set of differential equations resembling a linear spring-damper system:

$$\tau\ddot{\mathsf{y}} = \alpha_{\mathsf{y}}(\beta_{\mathsf{y}}(g - \mathsf{y}) - \dot{\mathsf{y}}) + f(\mathsf{x}) \tag{3.34}$$

$$\tau\dot{\mathsf{x}} = \alpha_{\mathsf{x}}\mathsf{x} \tag{3.35}$$

With appropriate values for the coefficients, the *transformation system* of Equation (3.34) ensures y converges stably to the point attractor g. This is akin to a potential field that enforces motion at any initial point towards this target position.

The forcing term f in Equation (3.34) is a non-linear function that encodes the shape of the trajectory, and enables generating complex motions. It is a sum of N_{bfs} weighted basis functions:

$$f(\mathsf{x}) = \frac{\sum_{i=1}^{N_{bfs}} \psi_i(\mathsf{x}) w_i}{\sum_{i=1}^{N_{bfs}} \psi_i(\mathsf{x})}$$

where ψ_i is often chosen to be a Gaussian kernel function, and the weights w_i define the attractor landscape. These weights can be learned to theoretically produce any arbitrary smooth trajectory shape.

Equation (3.35) is termed the *canonical system*, and controls the evolution of the trajectory in the first system through phase variable x, which decreases from $\mathsf{x} = 1$ to $\mathsf{x} = 0$, at which $\mathsf{y} = g$. This makes the trajectory representation time-invariant and, since f depends on x, it is possible to scale the motion with respect to time by tuning time constant, τ.

Note that, in keeping with the original notation ([52]), the evolving variable and the phase variable in the canonical system are denoted by y and x, respectively (using a distinct sans-serif font).

Among the advantages of DMPs as trajectory representations is their compatibility with learning from demonstrations. By capturing a demonstrated movement's profile in the form of position, velocity, and acceleration values using techniques such as kinesthetic teach-in or visual motion capture, a DMP can effectively be 'learned'. This process involves extracting parameters y_0, g, and τ and using a supervised learning approach to approximate the forcing term required to reproduce the demonstrated trajectory. For this, locally weighed regression (LWR) is used to determine the basis function weights w_i that minimize the locally weighted quadratic error between the target forcing term and that of the DMP.

DMPs enable complex movement representations that do not require explicit planning by exploiting coupling effects of the non-linear system equations. A notable consequence of this not having to rely on trajectories generated by conventional planners, which are less predictable than demonstrated and subsequently learned motions. The flexible and compact formulation of DMPs also facilitates generalizability of motions and opportunities for learning derivative solutions to constrained reaching problems, for example, by tuning their meta-parameters.

Other trajectory representations, that could be employed for motion reproduction and learning,

include Gaussian Mixture Models (GMMs) ([21]), Hidden Markov Models (HMMs) ([64]), Probabilistic Movement Primitives (ProMPs) ([81]), Interaction Primitives ([8]), and invariant trajectory representations ([113]).

3.3 Gaussian Processes

This section contains a brief description of Gaussian Processes and Gaussian Process Regression, due to their significance in model-based RL algorithms and other similar approaches dealt with in this report. For a more complete treatment of this topic, the reader is directed to the dedicated book by Rasmussen and Williams [116].

A Gaussian Process (GP) is a non-parametric model that defines a probabilistic distribution over functions, and is often used as a universal function approximation method. As a type of stochastic process, a GP effectively generalizes a multivariate Gaussian distribution to infinite dimensionality, and is formally defined as a collection of normally distributed random variables, any finite number of which have a joint Gaussian distribution. This formulation allows defining a GP over a time index, for example, and modelling uncertainty over a function $f(x)$ of some observable variable x as a distribution over all possible functions. A Bayesian approach can then be applied to make inferences on the form of the underlying function by updating the posterior GP with observations of x.

Similar to a multivariate Gaussian distribution, a GP is fully represented with a mean function and a covariance function, noting the generalization to the space of *functions*:

$$m(x) = \mathbb{E}[f(x)] \tag{3.36}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{3.37}$$

where the conventional symbol m avoids confusion with mean vector μ, and k is used to indicate the use of kernel functions for the covariance.

Intuitively, m defines the average shape of the underlying functions, while $k(x, x')$ defines the covariance between any two values of the function.

The covariance kernel function is a central element in GPs, since it effectively encodes assumptions on the characteristics and form of the underlying function, such as smoothness properties, and thus defines the nature of the search space. It enables representing the bounds of uncertainty in any region of the distribution of functions, which can be interpreted as the assumptions we can make on the shape of the function according to the amount of data observed around that region. This property is what facilitates an efficient Bayesian inference procedure, where the GP prior is iteratively updated with information contained in successive observations.

The prior of a GP is most often represented by a zero mean function, $m(x) = 0$ enforcing no initial assumptions on the general function landscape. The covariance function is usually defined in terms of a squared-exponential kernel function:

$$k(x, x') = exp(-\frac{1}{2}|x - x'|^2) \tag{3.38}$$

For the purposes of model identification and modelling dynamic systems, GPs are used to approximate some function f, while Gaussian Process Regression (GPR) is used for inferring the underlying function. In a Bayesian formulation, the posterior GP is continuously estimated, given new training data $x*$, by making predictions on the function distribution. By conditioning on these observations, GPR computes the mean and covariance functions, for a test point $x*$ and target value y, with the equations:

$$m_*(x) = \mathbf{k_*}^T K^{-1} y \tag{3.39}$$

$$k_*(x, x') = k(x_*, x_*) - \mathbf{k_*}^T K^{-1} \mathbf{k_*} \tag{3.40}$$

Here, $\mathbf{k_*}$ refers to the vector of covariances between $x*$ and the rest of the (stored) training points, while K is a matrix of the covariances between all pairs of training points (refer to [116])

GPs can be useful in robotics, particularly in the field of model-based reinforcement learning, for approximating robot and/or task dynamics. This then provides a predictive forward model that enables learning off-line and transferring learned policies to the robot, with the aim of reducing system interaction time and more data-efficient learning in demanding problems. The GPR procedure, however, can be computationally heavy, since the prediction step necessitates an expensive matrix inversion step. Nevertheless, recent approaches have enabled alleviating the computational burden using sparse approximations, local GPR, and utilizing GPUs, for example ([14]).

4

Methodology

The primary objective of this project is to evaluate an apprenticeship learning procedure that enables a robot to acquire a context-aware, human-oriented skill. This necessitates two essential analyses: i) validating the feasibility of a chosen algorithm for attaining the desired behaviour and its quantitative performance details, and ii) a human user study from which empirical conclusions on the success of the emergent behaviour can be drawn.

This chapter deals with the methodology followed for evaluating the implementation in both respects. Section 4.1 provides a rudimentary description of the procedure underlying this study. It also lays the foundations for subsequent result analyses, and the criteria used to evaluate both the implemented learning procedure as well as the resultant behaviour observed on the robot through experiments. Section 4.2 describes the actual user study with which a learned context-aware skill is evaluated qualitatively, including the experimental design and procedure.

4.1 Set-up

As previously mentioned, the human-robot object hand-over skill was chosen as a use-case to investigate the implications of reinforcement learning-based contextualization of a demonstrated task. In this project, the hand-over task is performed by the Toyota Human Support Robot (HSR), shown on Figure 4.1[1]. The dyadic task is performed unilaterally, with the hand-overs strictly being from the robot to the human, such that we focus on the robot's behaviour as it proactively transfers the object. In anticipation of contextual variations, we attempt to equip the robot to be contextually aware so that hand-overs are adjusted to the posture of the person (standing vs. seated) or the presence of an obstacle, for example. This is eventually tested in a lab mimicking a domestic setting, where subjects are offered an object by the robot in varying scenarios, in order to evaluate the impact of the contextual awareness.

We now briefly describe the methodology of the project.

Acquiring A Demonstration Chosen both as a prototypical human-robot collaborative task and for its simplicity and atomic nature, the robot hand-over skill is formalized as the movement of a grasped object to a chosen hand-over position in 3D space, following a chosen end-effector trajectory, with the aim of transferring the object to a person. These characteristics of the execution are preferably obtained

[1] https://mas-group.inf.h-brs.de/

from demonstrations, which makes dynamic movement primitives (DMPs) a natural choice for encoding the underlying movement. Not only do DMPs enable reproducing human demonstrations, they are also conducive to alteration by learning through their tunable parameters: trajectory shape weights w, goal position g, and time constant τ. Therefore, a DMP representation is used to capture a hand-over demonstration movement, as an initial step.

Defining Contexts The contexts which the robot can identify and subsequently adjust its behaviour to are defined in terms of factors that appreciably influence the optimal way with which an object can be handed over. For the purposes of our study, we consider the hand-over contexts relating to the posture of the receiver, the presence or absence of an intermediate obstacle, and the fragility or an inherent safety-critical property of the object.

Consequently, we symbolically define the context of a situation, $\mathcal{C} = \{c_1, c_2, ..., c_n\}$, as a set consisting of observed values of context parameters, c. In our setting, these parameters and their values can be defined as follows:

- Posture: $c_1 \in \{standing, seated, lying\_down\}$
- Obstacle-free: $c_2 \in \{True, False\}$
- Object-fragility: $c_3 \in \{True, False\}$

Figure 4.1: The HSR. Courtesy of H-BRS, MAS Group.

As an example, an observed hand-over context $\mathcal{C} = \{seated, True, True\}$ can be described as *handing a fragile object over to a seated person, with an obtrusive obstacle in the way.* With this information, the robot could make use of learned contextual policies to ensure its execution is tailored to these details.

Establishing Learnable Parameters Having formulated a representation of a hand-over movement in terms of parameterized DMPs, and defined our context parameters, we establish the relationship binding the two in our subsequent learning procedure. In particular, we conceptually determine the parameters of a DMP which predominantly influence an aspect of the hand-over execution, and thus affect a particular dimension of context. These DMP parameters are chosen to match the respective context parameters, such that choosing their values appropriately can facilitate an execution that is most adequate for the current context (value). To that end, we draw some parallels between our sets of parameters in the following.

The optimal hand-over position, dictated by the DMP goal position (or state) parameter g, depends on the current posture of the receiving person, indicated by context parameter c_1. This assertion stems from the intuitive fact that a person standing upright, sitting down, or lying down would have substantially different preferences for the optimal point at which they are expected to receive an object. The final hand-over position becomes a significant factor in the comfort of the person and the amount of effort they must expend, as well as how natural the action is perceived to be. For instance, a hand-over study conducted by Koay et. al. in [34] revealed that seated subjects significantly preferred a hand-over position

adequately adjusted to reflect their posture, over an arbitrary one, as one of the study's more prominent findings.

The shape of the hand-over trajectory, which is governed by the DMP basis function weights w_i, assembled in matrix W, may have less importance from the perspective of receiver posture, but can be a factor in distinguishing favoured behaviour for a variety of context factors. Among these is the presence or absence of an obstacle between the robot and the receiver (c_2), which could include table-tops, counters, or significantly large obstacles. Since the presence of such an obstacle is usually best avoided by following a different trajectory and may not require changing the final hand-over position, a contextual policy that generates appropriate DMP weight values to enable context-dependent trajectory shapes is a suitable solution. Note that the focus here is not on conventional obstacle avoidance, which could be accomplished through reactive planning, but rather on the possibility of learning various trajectory shapes and encoding them in a contextual policy.

The execution speed of a trajectory generated using DMPs can be tuned by changing time constant parameter τ. An application which relates to our third context parameter c_3 lies in changing the speed of the end-effector during a hand-over to account for certain properties of the object. These include fragility or possible spillage of a drink, for which the speed of the original demonstration may be hazardous or inconvenient.

To summarize, each context parameter is associated with a contextual policy which is to be learned for a particular DMP parameter, such that for:

- receiver postures, c_1: optimal hand-over/goal positions are learned, while trajectory shape and execution speed are constant;
- obstacle presence/absence, c_2: optimal trajectory shapes are learned, while hand-over/goal position and execution speed are constant;
- object fragility, c_3: optimal execution speeds are learned, while hand-over/goal position and trajectory shape are constant.

Learning Policies Our apprenticeship learning approach aims to extend a demonstrated, static hand-over policy to a contextually adaptive one, using a suitable reinforcement learning algorithm. The choice of said algorithm must invariably be preceded by formulating a policy representation, which would allow the robot's action to be contextually adaptive along different 'dimensions' of context. The consulted literature (refer to chapter 2) suggests that encapsulating a behaviour that adheres to multiple contexts of a single type is a challenge in its own right, although it is possible through probabilistic policy models such as GMMs, for example. Multiple context types/parameters can potentially be tackled through multivariate variants of such models, for example, but the multi-modality associated with each parameter is expected to complicate capturing the desired complex behaviour in a single model. *Hierarchical policy formulations have been identified as a promising approach, and are thus pursued in this work.*

We use probabilistic models and a hierarchical structure of multiple such policies: the former for their desirable properties including amenability to sequential improvements and direct encoding of

exploration, the latter for their suitability for multi-contextual learning. Namely, we employ three separate policies, one for each learnable DMP parameter (g,w, and τ), then simply aggregate their outcomes. The result is a coherent hand-over trajectory which takes multiple context parameters into consideration, thanks to the compact and compliant DMP representation.

The choice of algorithm is then determined based on its fulfilment of the requirements imposed by the policy formulation, such that the robot learns hand-over executions that are sensitive to each of c_1, c_2, and c_3. Appendix A contains a brief comparison of candidate algorithms, while Chapter 5 presents the choice algorithm in more detail.

Evaluating Algorithm Performance and Learned Policies The subsequent learning phase, similar to any machine learning procedure, requires careful analysis of an algorithm's performance with respect to a variety of tunable learning parameters. These include the number of iterations in which the policy (or, more generally, the 'model') is updated, the expended time and memory resources, and the dimensionality of inputs. In addition, factors like generalization performance, convergence properties, and sensitivity to the focal features in the data are similarly pertinent. For a policy search (PS) method, and reinforcement learning (RL) in general, it becomes necessary to also consider criteria such as reactions to different reward functions, the number of roll-outs per iteration, exploration parameter characteristics and values, policy complexity, and so on.

Apart from analysing the algorithm's performance across multiple parameters, we evaluate the quality of the learned policies, in terms of the behaviour manifested on the robot as it follows them. Criteria which are particularly significant for our case include how well a policy's produced actions resemble the desired, such as how close its sampled hand-over position or trajectory are to the preferred ones, and its ability to handle multiple context values: how it copes with encoding more than one desired action, and reliably choosing the right one for the current context. These preliminary evaluations are performed prior to actually observing behaviour on the robot, through simulations of policy outputs.

A description of these evaluations is provided in sub-section 4.1.1.

Applying Learned Policies on a Robot Since our learning procedure is performed in a form of simulation, along with the aforementioned preliminary evaluations, the next step involves transferring the learned policies to the robot, the HSR, which would be utilizing them. Sub-section 5.3.1 includes a description of the hand-over action implemented on the robot, on which different policies could be tested. The implementation would utilize a DMP formulation to execute hand-over trajectories, thus allowing us to apply policies parameterized by each of the context parameters, and learned for each of g, w, and τ. In essence, the set of parameters picked by each respective policy is sent in a collective command to the robot, and the resultant execution is expected to incorporate the contextual knowledge associated with each, in a manner similar to what was observed in simulations.

Experimentally Validating Merits of Learned Policies The final phase of the procedure followed in this study aims to conclusively validate the benefits of a context-aware approach to a hand-over action, as enabled by the learned policies. Our basic hypothesis stating that these considerations do improve upon

a robot-centric approach, which disregards the scenario the user is currently in, is verified in a human user study. The experiments involve the two 'behavioural modes' being used to execute hand-overs in varying scenarios, and their respective impacts on users being estimated through questionnaires. The conclusion arrived at as a result of this study would then provide confirmation of how promising an avenue apprenticeship learning is for the problem of contextual adaptation, and its merits.

Sub-section 4.1.2 describes the bases on which the learned behaviour will be evaluated in experiments, while section 4.2 explains the proposed experimental procedure.

4.1.1 Evaluating Learning Performance

The performance of the selected PS algorithm as it learns appropriate policies is systematically evaluated using quantitative and qualitative analyses. Naturally, this process is performed both iteratively and as a final assessment of performance on simulations. The learning procedure is driven by the insights gathered through observing the effects of parameter changes on quantitative performance measures, as well as perceived policy quality. Subsequently, the results achieved by what is considered the best set of parameters are then analysed to make conclusions about the characteristics of the algorithm. Particularly important criteria to examine include:

- Quality of resultant behaviour (policy outputs)
- Time needed to achieve said behaviour
- Number of policy update iterations, N
- Number of roll-outs per iteration, M
- Exploration parameter values
- Type and dimensionality of context vectors
- Number of DMP basis functions
- Reward functions
- Scalability
- Learning limitations
- Computational resources

The quality of the policy produced by the algorithm can be measured by how closely it resembles the desired behaviour. Since RL methods encode this into reward functions, our final achieved reward provides an intuitive and primary measure of the quality of the policy. Nevertheless, there are two important considerations to take into account. Firstly, while exploration parameters enable learning, their inclusion in final assessments produces unnecessary fluctuations in final reward values, especially if their values are significant. Therefore, they are set to zero when evaluating final policies. Secondly, our multi-contextual policies are expected to satisfy multiple modes of behaviour, depending on the observed context, which makes it necessary to evaluate policies based on rewards obtained for each possible value of a context parameter.

Determining the actual time required for the algorithm to converge on desirable behaviour is tantamount to estimating the required number of policy iterations. However, it is worth noting in its

own right when considering that learning may eventually be online, i.e. performed on the system directly, which is conceivable for robotic systems.

As with most machine learning procedures, the number of iterations required to attain a desired result is a key factor and a principal measure of efficiency.

When learning through reinforcement, it is often the case that policies are updated at each iteration after observing the result of multiple roll-outs: atomic episodes or independent executions of some task. This allows an algorithm to observe various possible outcomes of its exploration, and thus make a more informed update of the policy. Consequently, among the sought parameters is the number of roll-outs per iteration that achieves a good compromise between computational efficiency and resultant behaviour: an additional degree of freedom to our problem.

Exploration drives learning by reinforcement, and PS algorithms may employ substantially different exploration strategies, ranging from simple added Gaussian noise to information-theoretic policy updates. In all cases, the values of the parameters driving this exploration play a major role in deciding how well an algorithm performs, and thus warrant careful consideration.

Owing to the fact that we learn contextual policies, parameterized by context variables/vectors, the type and dimensionality of these parameters may significantly influence our ability to capture desired behaviour. Although lighter and less restrictive representations (e.g. using scalars and real numbers, respectively) are preferable, it is expected that certain algorithms and policy formulations may require more elaborate structures. An algorithm is thus also evaluated based on its suitability for different context representations.

The use of DMPs for capturing and reproducing trajectories also introduces additional design parameters, chief among which is the number of basis functions in the forcing term, which controls the precision of trajectory shape representations.

The critical role of reward functions and reward shaping in RL cannot be overstated: reward signals are the robot's only indication of its performance and latent objectives, and their characteristics ultimately decide the feasibility of learning some policy. Factors such as reward sparsity and degree of differentiation between outcome desirability are to be taken into account when designing reward functions. When coupled with policy update strategies employed by the chosen algorithm, we are then faced with the task of experimenting with various reward formulations to identify one that achieves the best results. (Chapter 6 includes the results of various reward functions used for respective policies in our work)

Scalability tests can provide insights into how well an algorithm scales with increasing parameter dimensionalities. For example, experimenting with different context parameter dimensions, or varying sizes of policy outputs are expected to reveal how well the algorithm may cope with different, possibly more complicated, problems.

The drastic differences in conventional RL algorithms and varieties of policy formulations necessitates evaluating and estimating the limitations of the proposed approach in capturing desired policies. These include difficulties in learning that can be traced back to restrictions imposed by chosen formalisms, such as the inability to escape local performance optima, or to represent some complex behaviour. Identifying limiting factors allows us to establish the suitability of the proposed procedure to other

problems exhibiting certain recognizable traits. An exemplary case is that of a neural network policy employing linear activations struggling to achieve non-linearities in desired behaviour.

Lastly, computational expenses imposed by the algorithm of choice is a fundamental evaluation criterion. Therefore, the time and memory resources required to achieve the final result must also be taken into consideration.

4.1.2 Evaluating Learned Behaviour

Following our evaluations of learned policies in the simulations in which they were learned, we proceed to evaluate the behaviour on the robot, and subsequently verify the hypothetical improvements behaviour learned through apprenticeship learning makes over a more naive approach. In particular, the context-aware behaviour enabled by the learned policies is implemented in addition to its opposite: contextually non-adaptive executions of an action that disregard current context. The latter, which is the more common approach, usually consists of situation-independent, pre-set executions whose parameters do not adapt to contextual differences.

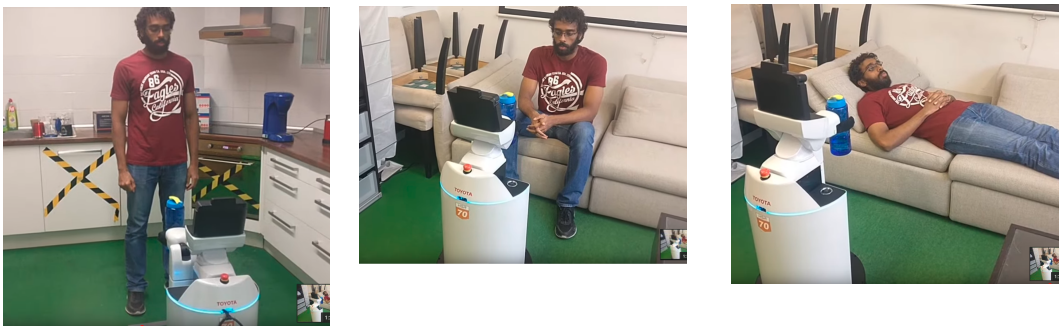
These evaluations are not performed by the person designing the learning procedure, in order to avoid undesirable biases; instead, we conduct a human user study to obtain subjective evaluations from multiple participants. In this study, we perform experiments with the objective of estimating typical users' perceptions of context-aware and context-unaware behaviours, such that we obtain conclusive evidence supporting or opposing our hypothesis. These experiments involve the robot performing hand-overs for participants in either case, before they fill out questionnaires that are later analysed in detail. In the first case, the robot is set to execute the same (predefined) motion in all scenarios; in the second, it samples executions from its learned policies, parameterized by observed context variables.

The questionnaire is designed to estimate a user's perception of the robot's performance, in terms of how appropriate its motions are and how comfortable it is to receive an object from it, since these are the ultimate factors governing a good hand-over. The criteria incorporated in the questionnaire include:

- Perception of observed behaviour
- Suitability for chosen contexts
- Naturalness of executions
- Predictability of executions
- Comfort and expended effort

Each kind of behaviour (context aware vs. unaware) is tested on these criteria in separate sections on the questionnaire, with a final assessment of which is generally preferred. Additionally, the questionnaire includes a section allowing users to provide descriptions and general comments. This is expected to help in identifying factors initially not taken into account and certain trends that may be informative for subsequent analyses. The questionnaire used in the experiments can be found in Appendix B.

In order to ensure maximal soundness of the conducted experiments, some additional measures are taken into account. Since the experiments are performed with a robot in a lab setting, volunteers with varying degrees of experience with the robot are recruited for the experiment. This ensures breaking



(a) Standing up (b) Sitting down (c) Lying down

Figure 4.2: Different postures assumed by participants in the HSR hand-over study

any habituation effects, which may affect the expectations and opinions of individuals that regularly interact with the robot. The order in which behaviour modes, as they are referred to in the questionnaire, are presented to a participant and the order of the sequence of predefined context scenarios are both randomized, to avoid any latent bias. Biases in opinion, in general, are also avoided by providing participants with identical experiment instructions and input, and conducting user trials in isolation from other participants.

The next section provides an explanation of the experiment procedure that was eventually designed for this study.

4.2 Experimental Design

This section contains descriptions of the experiment(s) designed to evaluate the performance of the learned hand-over skill in different contexts, and to verify whether it improves over a context-unaware variant of the same action.

4.2.1 Experiment Setting

Our experiments are performed using the Toyota Human Support Robot (HSR), in a quasi-domestic setting in our laboratory at Hochschule Bonn-Rhein-Sieg. The experiments were particularly designed to test a policy that chooses different hand-over positions, which was learned beforehand in simulations, such that the robot adapts hand-overs according to user posture (c_1). This is compared to the aforementioned contextually non-adaptive policy. The setting included three pre-determined positions at which a person could be situated, either standing up, sitting down, or lying down (on a couch). Figure 4.2 shows the respective positions and postures that are assumed by a participant during the experiment.

4.2.2 Experiment Procedure

A single experiment in the study was structured as follows. A sequence of three hand-overs, each in one of the contexts, is performed twice: once under the context-aware behaviour mode and once under

the context-unaware behaviour, for a total of six trials. Initially, the order in which the two behaviour modes are chosen and the sequence of contexts for a particular behaviour mode are both randomized. The first phase, consisting of the three hand-overs executed under the first behaviour mode, is followed by nine questions that can be answered on a 5-point Likert scale ranging from strong agreement to strong disagreement, and which test the initial perception of the user. Following the second phase, the same set of questions must be answered, this time for the second behaviour mode. Finally, the participant provides an overall preference over the behaviour modes on a 5-point Likert scale, and answers a set of open questions.

Each participant is briefed on the nature of the experiment and the main procedural details before their run, and then allowed to read the questionnaire in full. The experiment then starts, with each trial executed in the pre-determined sequence. If the participant is unsure of their opinion of a trial, or would like to reinforce it, they are allowed to ask for a repeat.

At each trial, a person detection software module is first used to detect a person in front of the robot, which is already holding an object in its gripper. When successful, the robot utilizes a posture identification module to estimate the person's current posture. Accordingly, it chooses a hand-over position, according to its policy, at which it presents the object to the person and awaits their reception. A reception detection component then utilizes the wrist force sensors on the robot to determine whether the person is trying to pull the object from its grasp, at which point it releases it and returns its arm to a neutral position. During these experiments, the robot executes a trajectory of the same shape for all trials.

We provide a video demonstrating the hand-overs performed during the experiments[2].

At the end of the experiment, the participant is allowed to fill out the final section of the questionnaire.

4.2.3 Result Analysis

Following collection of data from the participants of the HSR hand-over experiments, we then analyse the results to extract useful insights. As previously mentioned, each question can be answered in a scale from 1 to 5, which can be interpreted as a score range of $(-2, 2)$. Since each question in the set of nine which were posed in the questionnaire about each behaviour mode determines some aspect of the user's perception of the hand-overs executed by the robot (naturalness, suitability, etc.), it is possible to aggregate the scores provided by all users per question. A comparison between the total scores on each aspect for each behaviour mode can then reveal the extent to which one is perceived to be better than the other. (Refer to Appendix B for the full questionnaire, and chapter 7 for the list of posed questions)

In the same manner, the overall preference for the context-aware behaviour or its antithesis can be determined from the scores on the general conclusive question.

Finally, we analyse the general comments provided by the participants, in order to extract meaningful and possibly overlooked aspects of the hand-over and overall performance of the robot.

[2] www.youtube.com/watch?v=GguFJ2a7O6E

5

Solution

This chapter introduces the solution proposed for the problem of learning generalizable, context-dependent hand-over executions. In this project, as described in our methodology, we pursue an apprenticeship learning approach, in which we combine learning from demonstration and a model-based reinforcement learning algorithm. The choice of model-based policy search is motivated by the desire for a more data-efficient approach to learning a robot skill, such that we can minimize interaction time and, consequently, safety hazards, system wear, and operating costs.

Appendix A contains a brief overview and comparison of four candidate algorithms. Following a basic analysis of the properties of each, the proposed algorithm is presented in section 5.1 and elaborated on in finer detail. This includes an extensive explanation of the algorithm, its properties, notable merits, and foreseeable challenges. Finally, section 5.2 exposes details of the actual workflow and implementation details, including a formalization of the hand-over skill, the complete apprenticeship learning process, and a brief note on some of our notable improvements on existing implementations.

5.1 Proposed Algorithm

For this work, we have proposed the **C-REPS** algorithm as a suitable solution for the problem of learning a generalizable hand-over skill on a real robot. The brief comparison of four candidate algorithms presented in Appendix A suggests that the GPREPS algorithm is most promising for model-based contextual policy learning, and possesses various relevant properties. Although the algorithm traditionally uses GP predictive models, we have opted to avoid the computational and general complexity of Gaussian processes in favour of a significantly simpler notion of a forward model which is enabled by the DMP implementation (*pydmps*) and pre-determined reward functions. The result is a variant of C-REPS that learns policies in simulations, as opposed to the conventional model-free version. This section describes the algorithm in detail.

Contextual Relative Entropy Policy Search is a PS algorithm that can learn contextual policies in a hierarchical structure and performs information-theoretic policy updates. The algorithm employs a learnable stochastic upper-level policy and a deterministic lower-level policy; the former learning parameters that make the latter applicable to different contexts of a task. It was introduced by Kupcsik et. al. in [67] and [66], which contains a thorough explanation of its underlying principles, as a contextual

extension to the original REPS ([89]), such that restarting learning or capturing separate policies for every new task situation is avoided by following a more holistic approach to learning robot skills, for example.

The procedure of the C-REPS algorithm, methodically described in the following is almost identical to that of GPREPS, presented in section A.3 (of Appendix A) and summarized in Algorithm 3, apart from forward model learning which, for now, is abstracted away, by leaving the source of roll-outs (system or model) unspecified.

The general problem of contextual PS involves adapting the parameters of a parameterized policy ω to an observed context c, and C-REPS solves this in the most efficient way: using a hierarchical policy decomposition, which avoids the tedium of the alternative, in which a separate policy for each context must be learned.

Hence, we define the two constituents:

- Upper-level policy $\pi(\omega|c)$, implemented as a linear-Gaussian model
- Lower-level policy $\pi(u|x,\omega)$, implemented as dynamic movement primitives (DMPs) , in our case

The upper-level policy chooses parameters ω that influence a lower-level policy, given some context parameter c, and is implemented as a linear-Gaussian model:

$$\pi(\omega|c) \approx \mathcal{N}(\omega|a + Ac, \Sigma) \tag{5.1}$$

Its implementation as a parametric model makes sampling ω for artificial roll-outs particularly simple. The parameters of the policy $\theta = \{a, A, \Sigma\}$ represent the mean and covariance of the parameter distribution, the latter directly encoding the exploration required for learning. Context variable c is assumed to be sampled from a distribution defined by $\mu(c) = a + Ac$.

The lower-level policy executes the actual system trajectories, conditioned on parameters ω throughout the current episode. Since this is often on a robot, its parameters consist of low-level control signals u and current state variable x (following notation adopted for RL in robotics), which depend on the current timestep during execution. We use DMPs as lower-level policies. Technically, they do not represent the lowest level of execution, since they define an end-effector trajectory and are thus succeeded by an inverse kinematics controller that determines the joint commands required to move along that trajectory. Nevertheless, this controller is abstracted away as it bears no direct impact on the learning algorithm's procedure.

The expected return of an execution (or episode), $R_{c\omega}$ is determined by the expectation over all trajectories, τ, of the output of reward function $\mathcal{R}(\tau, c)$ conditioned on the current context and sampled lower-level policy parameters ([67]):

$$R_{c\omega} = \mathbb{E}_\tau[r(\tau, c)|c, \omega] = \int_\tau p(\tau|c, \omega)\mathcal{R}(\tau, c)d\tau, \tag{5.2}$$

where $p(\tau|c, \omega)$ denotes a distribution over trajectories.

The problem of finding optimal upper-level policy parameters θ then amounts to maximizing the

expected return over the distributions of context and lower-level parameters (the upper-level policy):

$$J_\pi = \int_c \int_\omega \mu(c)\pi(\omega|c)R_{c\omega}dcd\omega \tag{5.3}$$

which is identical to Eq. (3.8), except that actions a are substituted for the output of the upper-level policy: ω.

The C-REPS approach defines and optimizes over a trajectory distribution that constitutes a joint probability of the context and parameter variables: $p(c,\omega) = \mu(c)\pi(\omega|c)$, which relaxes an assumption of the availability of many ω samples for every context vector c ([35]). As a consequence, an added constraint to the subsequent optimization procedure is: $\forall c : \int_\omega p(c,\omega) = \mu(c)$, while the main objective is reduced to maximizing:

$$J_\pi = \int_c \int_\omega p(c,\omega)R_{c\omega}dcd\omega \tag{5.4}$$

The above constraint on the context distribution results in infinitely many constraints for continuous context variables, which the authors of C-REPS address by matching feature expectations (averages) instead of single probabilities, expressed with:

$$\int_c \int_\omega p(c,\omega)\phi(c)dcd\omega = \hat{\phi} \tag{5.5}$$

introducing a context feature vector $\phi(c)$ and its mean $\hat{\phi}$, which is taken to be the average over observed contexts.

An integral part of REPS algorithms are information-theoretic policy updates, which come in the form of an additional constraint that requires relative entropy bound parameter ϵ. Policy updates are restricted so that the relative entropy (i.e., Kullback-Leibler divergence) between consecutive trajectory distributions is bounded by ϵ in order to minimize excessively greedy policy updates and information losses:

$$D_{KL}(p(c,\omega)||q(c,\omega)) = \int_c \int_\omega p(c,\omega)log\frac{p(c,\omega)}{q(c,\omega)}dcd\omega \leq \epsilon, \tag{5.6}$$

where $q(c,\omega)$ refers to the trajectory distribution of the previous policy iteration, i.e. the old policy. It is also possible to use multiple previous policies and aggregate their results in $q(c,\omega)$.

In addition to the fact that the trajectory distribution must be a valid probability distribution, the above factors together define the central constrained optimization problem ([66]):

$$\begin{aligned} \max_{p} \quad & \int_c \int_\omega p(c,\omega) R_{c\omega} dc d\omega \\ \text{s.t.:} \quad & \int_c \int_\omega p(c,\omega) log \frac{p(c,\omega)}{q(c,\omega)} dc d\omega \leq \epsilon \\ & \int_c \int_\omega p(c,\omega) \phi(c) dc d\omega = \hat{\phi} \\ & \int_c \int_\omega p(c,\omega) dc d\omega = 1 \end{aligned} \tag{5.7}$$

where the trajectory distribution chosen so as to maximize J_π implies an underlying optimal upper-level policy.

Keeping in mind that $p(c,\omega) = \mu(c)\pi(\omega|c)$, the optimal upper-level policy can be solved for using the Lagrangian formulation:

$$\pi(\omega|c) \propto q(c,\omega) \exp\left(\frac{R_{c\omega} - V(c)}{\eta}\right) \tag{5.8}$$

This equation introduces context-dependent baseline in the form of derived 'value function' $V(c) = \widetilde{\theta}^T \phi(c)$ and Lagrangian parameters η and $\widetilde{\theta}$ which account for the second and third constraints in Eq. (5.7), respectively. These parameters are determined by optimizing the convex dual function:

$$g(\eta, \widetilde{\theta}) = \eta log \left(\int_c \int_\omega q(c,\omega) \exp\left(\frac{R_{c\omega} - V(c)}{\eta}\right) dc d\omega \right) + \eta\epsilon + \widetilde{\theta}^T \hat{\phi} \tag{5.9}$$

This optimization problem requires a data-set of N simulated or real trajectories containing respective observed contexts, sampled parameters, and received or estimated rewards: $\mathcal{D} = \{c^{[i]}, \omega^{[i]}, R_{c\omega}^{[i]}\}_{i=1,\ldots,N}$ (as outlined in Algorithm 3). Using these samples of the old trajectory distribution (and thus, policy) $q(c,\omega)$, the integral in Eq. (5.9) is approximated by a summation. Each sample in $\mathcal{D}$ is then given an importance weighting, given by the probability of each:

$$p^{[i]} = \exp\left(\frac{R_{c\omega}^{[i]} - V(c^{[i]})}{\eta}\right) \tag{5.10}$$

The final step of the algorithm involves using these sample weights to re-estimate the parameters, θ^*, of $\pi(\omega|c)$: a, A, and Σ. The choice of a parametric policy model enables using a simple weighted maximum likelihood estimation (WMLE) policy update rule. As shown in [66], this can be computed with the following equations, where S is a $N \times D_c$ context matrix with rows of the form $[1, c^{[i]}]$ (where D_c is the dimensionality of context vector c), Ω is a $N \times b$ parameter matrix whose rows contain sampled parameters ω (D_ω denotes the dimensionality of parameter vector ω), and P is a $N \times N$ diagonal matrix containing sample weightings $p^{[i]}$:

$$\begin{bmatrix} \mathbf{a}^T \\ \mathbf{A}^T \end{bmatrix} = (S^T P S)^{-1} S^T P \Omega \tag{5.11}$$

$$\mathbf{\Sigma} = \frac{\sum_{i=1}^{N} p^{[i]} (\omega^{[i]} - \mu^{[i]})(\omega^{[i]} - \mu^{[i]})^T}{\sum_{i=1}^{N} p^{[i]}} \tag{5.12}$$

$$\mu^{[i]} = \mathbf{a} + \mathbf{A} c^{[i]} \tag{5.13}$$

Parameters $\{a, A, \Sigma\}$ are then used to update $\pi(\omega|c)$, in Eq. (5.1), and the algorithm normally iterates for the specified number of policy updates. An alternative approach (not explored here) can equip the algorithm to run until some performance measure is satisfied, as in the PILCO and Black-DROPS algorithms.

Exploration Characteristics As previously mentioned, the information-theoretic policy updates of C-REPS enforce staying close to the observed data, bringing stability to the learning process and a useful robustness measure to our apprenticeship learning approach. Bounding the relative entropy helps in striking a balance between forgetting (as a means for broader exploration) and experience retention. As a consequence, ϵ can be regarded as a factor in the exploration-exploitation trade-off, alongside the implicit exploration provided by the covariance of our stochastic upper-level policy, Σ. With reference to the performance criteria in sub-section 4.1.1, the two variables constitute our exploration parameters.

It bears mentioning that the incorporation of all trajectory samples, with their respective weightings $p^{[i]}$, in the final policy update can be significantly beneficial. Instead of taking one or a subset of samples that perform best into account, C-REPS attempts to extract the most information from all trials, including possibly unsuccessful but close executions, by simply weighting them according to levels of performance. Although this may conceivably slow down learning in some cases, it can help in systematically converging to the optimal parameters when navigating a tricky reward landscape. ϵ plays a major part in this step, controlling how discriminative the sample weightings are according to performance (higher values strongly favour 'successful' trajectories and drive the weights of less effective ones to zero; lower values offer a more relaxed criterion).

REPS algorithms encode a built-in correlated exploration strategy, similar to stochastic optimizers like CMA-ES, which is used in the Black-DROPS algorithm (refer to section A.2, of Appendix A). This implies that all elements of the policy model's covariance matrix are updated per iteration, unlike methods that over-simplify by only using and re-computing a diagonal covariance matrix, such as PILCO ([35]). The more realistic exploration, which takes parameter correlations into account, is expected to increase learning speed.

Finally, it is important to note that C-REPS, and REPS algorithms is general, may suffer from a problem shared by most stochastic search procedures: premature convergence on sub-optimal parameters. Updating the search, or exploration, distribution using the information-theoretic approach is expected to mitigate this, but it remains a problem: the policy search may still collapse into an undesirable point-estimate ([2]). In the next section (particularly sub-section 5.2.6), we describe a simple regularization

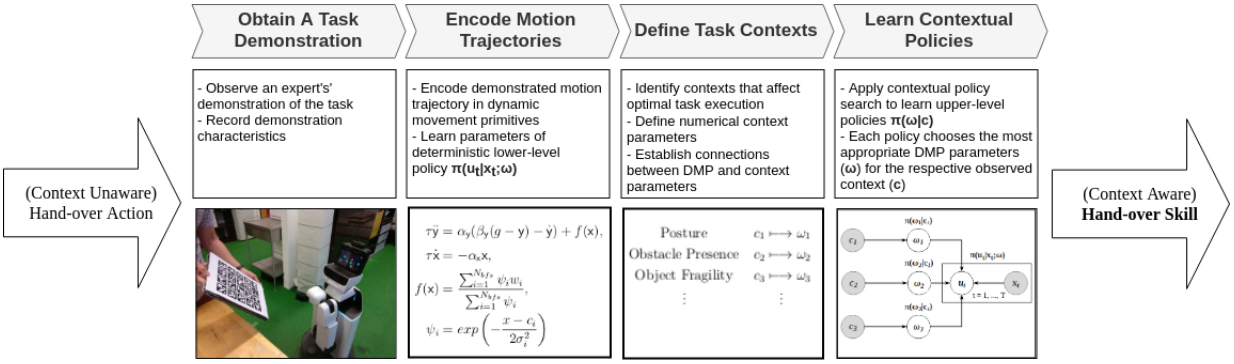


Figure 5.1: The main stages of our apprenticeship learning procedure for constructing a context-aware hand-over skill.

step we added to the algorithm that performs random restarts of Σ to avoid this problem, which we encountered in some cases.

5.2 Implementation

This section contains the details of how the hand-over skill was represented on the HSR, and how the apprenticeship learning process was implemented in order to achieve context-aware executions of object hand-overs. In the respective sub-sections, we describe the constituents of the hand-over skill, the process of acquiring demonstrations, capturing demonstrated trajectories, learning to generalize them to different contexts, executing context-aware hand-overs, and some notable contributions. Figure 5.1 portrays the general workflow of our implementation.

5.2.1 The Hand-Over Skill

The hand-over skill was implemented on the HSR, using ROS, as a finite state machine consisting of three states which constitute the phases of the action:

1. Detect people
2. Identify person posture
3. Hand object over

The robot is assumed to be holding an object in its gripper in a neutral position (with the gripper oriented forward away from the robot), as shown on Figure 5.2. Once commanded to perform the hand-over, the robot follows the phases enumerated above, such that it first attempts to detect a person, identify their posture, then execute an appropriate trajectory. The robot then awaits the person's reception of the object, which it detects using its wrist force sensor. For that purpose, we implement the CUSUM change detection algorithm.

Figure 5.2: The HSR grasping a bottle in a neutral position.

In the following, we describe each phase of the hand-over skill implementation[1] in some detail.

Detecting People

The first phase of the skill involves detecting the person to whom the object would be handed over. For this, images captured by the HSR's RGB-D camera are used as inputs to a Single Shot MultiBox Detector (SSD) model, trained on the COCO image dataset to detect people. The detector is implemented using the Keras library and is embedded in a ROS action server[2].

The associated detection action then returns a list of people that were detected in the images, along with a detection confidence score and their respective bounding boxes on the image, which are necessary for the next phase. At the moment, we consider the first detected person, often the one directly in front of the robot, the potential receiver of the object and thus the subsequent target.

Identifying A Person's Posture

The second phase, preceding the hand-over, involves identifying the posture of the person detected as the receiver of the object. This step enables the robot to autonomously determine the value of context parameter c_1 (referred to in section 4), which can assume one of the values: $\{standing, seated, lying\_down\}$. The posture context parameter is particularly chosen in the implementation due to its importance for subsequent experiments in the conducted human study.

Person posture identification is achieved using a simple heuristic that compares the height and width dimensions of the bounding box associated with the person. In particular, we determine the height-width ratio, κ , as the ratio between the number of pixels in the vertical and horizontal dimensions of the bounding box. *Using experimentally determined threshold values*, we define the posture identification decision function:

$$c_1 = \begin{cases} \text{lying\_down}, & \text{if } \kappa \leq 0.6 \\ \text{seated}, & \text{if } 0.6 < \kappa \leq 1.65 \\ \text{standing}, & \text{if } \kappa \geq 1.65 \end{cases} \tag{5.14}$$

Handing the Object Over

The main part of the skill, the hand-over action[3], requires the values of the context parameters (currently: c_1, as determined in the posture identification step, and c_2), and executes the motion of the end-effector that brings the object to the person in a suitable manner. The context parameters are used to sample lower-level policy parameters ω_1 and ω_2 from learned upper-level policies $\pi(\omega_1|c_1)$ and $\pi(\omega_2|c_2)$, which determine the hand-over (goal) position and the trajectory shape, respectively (sub-section 5.2.4 contains

[1] https://github.com/b-it-bots/mas_domestic_robotics/tree/feature/hand-over-action/mdr_planning/mdr_scenarios/mdr_demos/mdr_demo_context_aware_hand_over

[2] https://github.com/b-it-bots/ssd_keras_ros

[3] https://github.com/b-it-bots/mas_domestic_robotics/tree/feature/hand-over-action/mdr_planning/mdr_actions/mdr_manipulation_actions/mdr_hand_over_action

more details about these policies). Subsequently, these parameters are combined in a lower-level policy, implemented as DMPs of the form:

$$\tau \ddot{\mathsf{y}} = \alpha_{\mathsf{y}}(\beta_{\mathsf{y}}(g - \mathsf{y}) - \dot{\mathsf{y}}) + f(\mathsf{x})$$
$$\tau \dot{\mathsf{x}} = -\alpha_{\mathsf{x}}\mathsf{x}$$

We recall that the forcing term in the first equation is:

$$f(\mathsf{x}) = \frac{\sum_{i=1}^{N_{bfs}} \psi_i w_i}{\sum_{i=1}^{N_{bfs}} \psi_i},$$

and we use N_{bfs} Gaussian kernel basis functions of the form:

$$\psi_i = exp\left(-\frac{x - \mu_i}{2\sigma_i^2}\right)$$

For a more involved discussion of DMPs, refer to section 3.2, where the DMP equations have been introduced, and are repeated here for convenience. In our work, the DMP lower-level policy governs the evolution of end-effector position in three-dimensional Cartesian space, meaning we effectively use three dynamical system equations. The equations are written in compact form above, in which the target is vector $\mathsf{y} = \{x, y, z\}$, encapsulating all three coordinate variables.

For our hand-over skill, the DMPs are supplied with the parameters $\omega_1 = g$ and $\omega_2 = W$ (where W encapsulates the weights of all basis functions), and are then used to execute context-aware roll-outs of the action. In addition, the skill can be used with predetermined values (forgoing policy sampling) for both sets of parameters, which we use to simulate context-unaware hand-overs that execute the same motion in any scenario, regardless of identified person posture, for example.

The final part of the action involves detecting when to release the object to give it to the person, after stopping at the designated hand-over position. For this, we make use of wrist force sensors on the HSR to detect when the person is pulling the object with the intention of receiving it. According to an empirically determined threshold, the change in steady-state force readings is continuously evaluated using a change detection algorithm, CUSUM, which decides when the sensed forces imply that the person is actively pulling the object from the robot's grasp. This is discussed further in sub-section 5.2.6.

5.2.2 Obtaining a Demonstration of the Task

As explained in the discussion of our methodology in section 4.1, we initialize our learning from a human demonstration of the hand-over task, and capture it in a flexible motion primitive formulation. This constitutes the first part of apprenticeship learning, which we advocate as a promising approach to robot skill learning in general. Extracting these initial, pre-structured policies serves to speed up the learning process and allows exploiting human expert knowledge, with the aim of attaining learned contextual policies, and possibly even surpassing performance of demonstrations.

In order to teach the HSR a particular end-effector trajectory from demonstration, we use a motion capture approach. According to the LfD method categorization presented by Argall et. al. in [9], we rely on imitation through external observation, where the robot observes an external movement and uses its sensors (a camera) to record an executed motion for subsequent reproduction. Although this approach may potentially result in correspondence problems, it is relatively easy and less invasive compared to kinesthetic teach-in, which is also infeasible on some platforms.

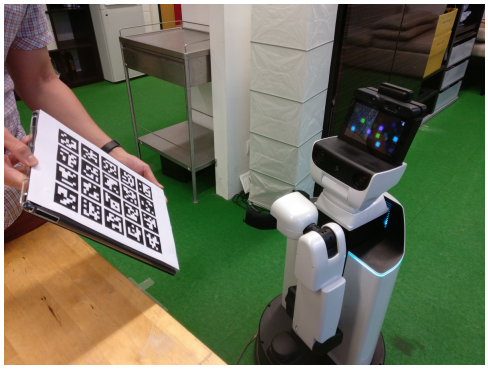


Figure 5.3: Demonstrating a trajectory to the HSR using a motion-capture imitation learning approach. Courtesy of Mitrevski et. al. ([76])

During the procedure, the robot is set to capture the motion of a marker that is moved in front of its camera (see Figure 5.3), recording the position, velocity, and acceleration values of the trajectory: $y_{demo}(t)$, $\dot{y}_{demo}(t)$, and $\ddot{y}_{demo}(t)$, for the duration of the motion. Subsequently, the DMP with which the trajectory can be executed by the robot is 'learned', which amounts to extracting its parameters g, y_0, τ, and W. The former three are straightforward to determine, while the last involves approximating the forcing term function using locally weighted regression (LWR). The supervised learning process is used to determine a weight value w_i for each basis function ψ_i, such that the error between the demonstrated trajectory and a roll-out, executed using the resultant DMP, is minimized.

For a more detailed explanation of the process of learning DMP parameters from a demonstration, the reader is directed to the comprehensive treatment of DMPs by Ijspeert et. al. in [52].

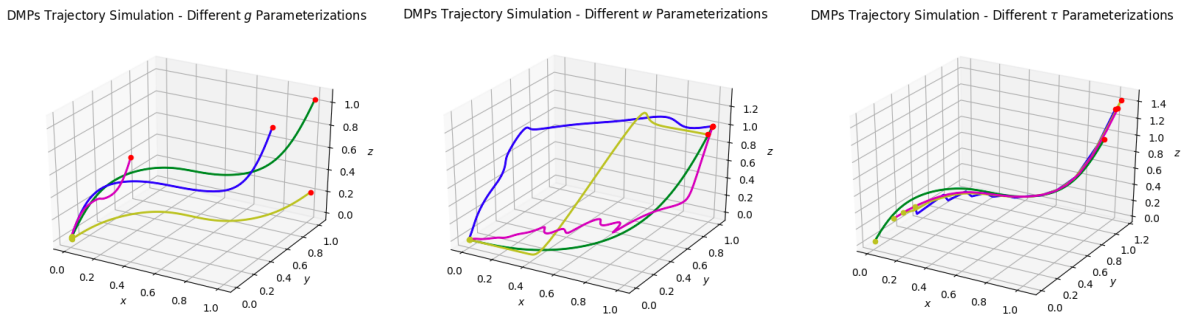
The adopted motion capture imitation learning procedure was presented by Mitrevski et. al. in [76], where a more detailed description of the procedure can be found.

5.2.3 Capturing Trajectories in Dynamic Movement Primitives

We use the *pydmps*[4] Python package to implement the trajectory DMPs. The package contains the functionalities required to represent trajectories in terms of DMPs, learn DMP parameters from a provided demonstration trajectory, and execute subsequent DMP roll-outs. The latter capability is especially instrumental in our learning-by-simulation approach, since it allows simulating a three-dimensional trajectory that can be achieved with a particular parameterization of the DMPs. Figure 5.4 shows three DMPs (in x, y, and z) producing four trajectories for different parameterizations of goal position g, shape weights W, and time constant τ, respectively.

For our purposes, this provides a convenient way to estimate the results (return) of a particular hand-over trajectory, and thus constitutes a 'predictive' model, such that we can predict the motion that would be executed by the robot. Using this to learn the skill in simulations, sometimes called *mental rehearsal* in the RL literature, makes our C-REPS implementation model-based rather than model-free, with the distinction from GPREPS being that we do not employ GP models.

[4] https://github.com/studywolf/pydmps



(a) Varying goal position g (b) Varying shape weights W (c) Varying time constant τ

Figure 5.4: 3D trajectories simulated using DMPs for different parameterizations

As a result of this process, we now have a deterministic lower-level policy, which generates the appropriate end-effector position at each time-step in the duration of a hand-over execution. (Again, this is in turn processed by a low-level inverse kinematics controller, to determine the necessary robot joint commands). In order to make this policy generalizable to different contexts, we learn the upper-level policies required to sample the parameters of the DMPs, ω, which result in the most appropriate trajectories. We discuss the underlying procedure next.

5.2.4 Learning to Generalize to Different Contexts

Following acquisition of initial upper-level policies from a demonstration and formulation of the lower-level policy to be used in our C-REPS algorithm, we discuss how the upper-level policies can be progressively improved to optimize altered executions of the task, conditioned on the current contexts. We base our implementation of C-REPS on that of Ricardo Dominguez: *PyCREPS*[5], which is directly based on the work of Kupcsik et. al. in [67] and [66]. Although PyCREPS employs the basic learning procedure of C-REPS, as described in the original publications, its policy formulations are geared towards RL benchmark problems found in the *OpenAI gym*[6] environment, such as the Cart-Pole and Acrobot problems. Due to this fact, in addition to some implementation inaccuracies, significant changes to the source code were necessary to effectively learn context-aware hand-overs.

In order to learn the contextual upper-level policy $\pi(\omega|c)$ responsible for sampling each DMP parameter ω, we first initialize the policy parameters to start from the value of the respective parameter obtained from the demonstration. Additionally, we initialize exploration parameters Σ to pre-determined values, in addition to ϵ, number of iterations N, number of roll-outs M, context vectors c, etc. For the lower-level policy, we clamp all parameters except ω to fixed values, and thus execute roll-outs that only depend on the upper-policy's ω, for policy evaluations. Clearly, each $\pi(\omega|c)$ is guided by a reward function that is shaped to facilitate learning values of ω that suit the values of c.

[5] https://github.com/RicardoDominguez/PyCREPS
[6] https://github.com/openai/gym

In the following we discuss two learned upper-level policies: one for learning adequate hand-over positions and another for trajectories, meant to handle posture and obstacle-presence contexts (c_1 and c_2, as defined in section 4.1) , respectively. Additionally, we briefly discuss the limitation that complicates learning meaningful context-dependent execution speeds for c_3.

Position-dependent Context Generalization

When trying to learn context-dependent hand-over positions using our algorithm, given a single demonstration, we attempt to learn an upper-level policy that samples $\omega = g$, given $c_1 \in \{standing, seated, lying\_down\}$, for example. The policy to be learned must therefore sample hand-over positions that fit the current, position-dependent context, and is given by:

$$\pi(g|c_1) = \mathcal{N}(g|a + Ac_1, \Sigma) \tag{5.15}$$

In order to initialize $\pi(g|c_1)$ from the demonstration, we ensure that the initial mean of the Gaussian distribution is equal to the hand-over position observed in the demonstration: $\mu(c_1) = g_{demo}$. If the demonstration runs for some duration T, g_{demo} is simply equivalent to the last recorded 3D position, $\mathsf{y}_{demo}(T)$. We then simply choose a to be equivalent to this vector: $a = g_{demo}$, and A to be a zero matrix of appropriate dimensions (which depend on the dimensionality of context vector c_1, and vector a). Covariance matrix Σ is initialized for equal, uncorrelated exploration: $\Sigma = I$, but may alternatively be set to arbitrary values (as will be proven in section 6.2.1, this initialization strategy is not ideal).

The learning process amounts to iteratively evaluating and updating parameters $\theta = \{a, A, \Sigma\}$ for N iterations.

At each iteration, a random value of c_1 is drawn, and used to sample a candidate hand-over position $g \sim \pi(g|c_1)$, which is provided to the lower-level policy. This policy, a system of three DMP equations, has trajectory shape, initial position, and time constant parameters fixed at the values extracted from the demonstration: $W = W_{demo}$ (such that: $\forall i \in \{1, ..., N_{bfs}\}, w_i = w_{i,demo}$), $\mathsf{y}_0 = \mathsf{y}_{demo}(0)$, and $\tau = \tau_{demo}$. Therefore, the only variable parameter, g, wholly governs the difference in performance between each consecutive roll-out, as determined by the reward function. We can thus assign credit or blame solely to the parameter values chosen by the stochastic upper-level policy.

The context-dependent reward function used to evaluate the choices of g is a Euclidean distance-based performance measure, which rewards an execution based on closeness to a set of empirically determined 'ideal' hand-over positions. Namely, we identified a set of position vectors $\hat{g}^1$, $\hat{g}^2$, and $\hat{g}^3$ that were determined to suit each respective value of c_1 best. The result is the following conditional reward function, which penalizes distance from the respective ideal position:

$$\mathcal{R}_g(g, c_1) = \begin{cases} \frac{1}{||g-\hat{g}^1||_2}, & \text{if } c_1 = \text{standing} \\ \frac{1}{||g-\hat{g}^2||_2}, & \text{if } c_1 = \text{seated} \\ \frac{1}{||g-\hat{g}^3||_2}, & \text{if } c_1 = \text{lying\_down} \end{cases} \tag{5.16}$$

It is worth noting that the crafted reward function, along with the experimentally determined ideal hand-over positions, constitutes some of the contextual knowledge incorporated into the robot's process of learning a human-oriented task: 'expert' knowledge of the approximate positions at which an object should be handed over, with regards to the receiver's posture. More details of the reward function, and other candidate distance-based measures, can be found in sub-section 6.1.2.

After executing M simulated roll-outs with current parameters θ, and recording the estimated rewards $R_{c\omega}$ of each, the algorithm computes each sample's weighting, $p^{[i]}$ by minimizing the function in Eq. (5.9), and using Eq. (5.10). The Lagrangian parameters are obtained by applying the L-BFGS-B minimizer on the dual function. With the sampled parameters g, their associated context variable c_1, and their calculated weightings, the WMLE equations (5.11-5.13) are then applied to update the upper-level policy's parameters at that iteration.

As the algorithm iterates, parameters $\theta = \{a, A, \Sigma\}$ are expected to shift appropriately such that reward $R_{c\omega}$ is simultaneously maximized for all the values of c_1. This can be observed in the simulated trajectories, with the final trajectory positions sampled for each context value progressively converging to $\hat{g}^1$, $\hat{g}^2$, and $\hat{g}^3$. Ideally, a and A would stabilize and the values of Σ would diminish as it becomes more certain of the values of sampled parameters that would yield maximum reward.

Trajectory-dependent Context Generalization

The second constituent of our hand-over generalization strategy involves learning context-dependent trajectories, governed by basis function weights in the forcing term of the DMPs used as a lower-level policy. Denoting the vector of these so-called trajectory shape weights by $W = [w_1, w_2, ..., w_{N_{bfs}}]^T$, the second case requires a separate upper-level policy that learns the optimal $\omega = W$ for each situation dictated by context parameter c_2.

As discussed in section 4.1, we have chosen c_2 to signify the presence or absence of an obstacle between the robot and the receiver that impedes normal execution of the hand-over, for simplicity, although one may conceive of various other contexts that may necessitate adaptive trajectory generation.

Similar to the previous case, we seek to learn a policy:

$$\pi(W|c_2) = \mathcal{N}(W|a + Ac_2, \Sigma) \tag{5.17}$$

which generalizes parameters W for the same lower-level policy: a set of three DMPs which dictates the trajectory of the end-effector in 3D space. However, when learning $\pi(W|c_2)$, the lower-level policy parameters are initialized differently in comparison to the case of $\pi(g|c_1)$. While $\mathsf{y}_0 = \mathsf{y}_{demo}(0)$ and $\tau = \tau_{demo}$ remain, the hand-over position is now also fixed to match the demonstration's: $g = \mathsf{y}_{demo}(T)$, and W is now the variable chosen by the upper-level policy. In much the same way, this allows us to evaluate the latter only based on the quality of the trajectory shape achieved by the sampled collection of weights, w_i. As a consequence, $a = W$ (ensuring W is a vector), A is a zero matrix, and $\Sigma = I$, as an initial choice.

The reward function used in this case is similar in concept to the first: finding a hypothetical

trajectory shape that best fits each context, and designing a reward signal that guides the algorithm into learning to approach these shapes in order to adapt to the requirements of each situation. Therefore, we provide context-specific 'ideal' trajectories in terms of position coordinates: $\hat{\mathsf{y}}^1$ and $\hat{\mathsf{y}}^2$, and then evaluate the simulated roll-outs by assigning the reward:

$$\mathcal{R}_{\mathcal{W}}(\mathsf{y}, c_2) = \begin{cases} \frac{1}{d(\hat{\mathsf{y}}, \hat{\mathsf{y}}^1)}, & \text{if } c_2 = \text{False} \\ \frac{1}{d(\hat{\mathsf{y}}, \hat{\mathsf{y}}^2)}, & \text{if } c_2 = \text{True} \end{cases}, \tag{5.18}$$

where $d(\cdot, \cdot)$ is some distance measure that allows us to assess how close the executed trajectory is to the desired. This is now left unspecified, but sub-section 6.2.2 describes the point-wise trajectory differences measure we eventually use, along with various candidates for $d(\cdot, \cdot)$. Note that this performance measure may also be in terms of weight vector W.

The process of learning $\pi(W|c_2)$ is identical to that of $\pi(g|c_1)$: we repeatedly sample c_2 and simulate M roll-outs, storing the values of c_2, sampled W's and the roll-out rewards $R_{c\omega}$. Then, the constrained optimization problem is solved using the L-BFGS-B minimizer to obtain the sample weightings $p^{[i]}$, which are then used in the WMLE policy update equations to find the optimal $\pi(W|c_2)$. This is repeated for the specified number of iterations, N.

With each update of upper-level policy parameters $\theta = \{a, A, \Sigma\}$, we expect to observe trajectories that approach the desired shapes, for each respective context (keeping in mind that the hand-over position is fixed during learning).

Comparative Complexity It is important to note, however, that learning $\pi(W|c_2)$ is significantly more difficult than learning $\pi(g|c_1)$ due to the dimensionality of the respective ω. In the case of $\pi(g|c_1)$, the policy output is a three-dimensional vector, which imposes the following respective sizes on a, A, and Σ: 1×3, $c \times 3$, and 3×3. Comparatively, the size of W is much larger, since it directly depends on the number of basis functions, N_{bfs}. For the same number of DMPs (3) and basis functions ($N_{bfs} = 150$) we use for $\pi(g|c_1)$, W is a vector of length 450. Therefore, the sizes of a, A, and Σ become 1×450, $c \times 450$, and 450×450, respectively, which clearly leads to a more complex problem.

As will be discussed in more detail in section 6.2, this problem becomes apparent when attempting to learn a policy whose parameters are expected to generalize over more than one context. Although the algorithm is capable of learning any single trajectory well, albeit with a significant number of policy iterations, it is currently unable to learn two simultaneously, using the same set of parameter θ. At the moment, we learn separate policies $\pi(W|c_2)$, one for each value of c_2, and sample from the appropriate one. Nevertheless, we hope to solve this problem in future pursuits.

Speed-dependent Context Generalization

As explained in our methodology in section 4.1, the hand-over skill can be made generalizable to different contexts along a third aspect of the motion: speed of execution. Namely, a third upper-level policy, $\pi(\tau|c_3)$, would learn optimal time constant values of the DMP lower-level policy, τ, to match the requirements

introduced by context parameter c_3. Since τ alters only the speed with which a motion is executed, it could theoretically be used to alter hand-over speeds to account for how fragile/hazardous the object being handed over is (c_3).

However, due to an implicit limitation in the DMP implementation provided by the *pydmps* package, it was not possible to produce trajectories that are meaningfully adjusted in speed of execution. As a result, no relevant results were produced for this generalization case, which is the reason the other two cases are the focus of this work. Appropriate speed-dependent context generalization of the hand-over skill is thus left for future work. More details on the results leading up to this conclusion are presented in section 6.3.

5.2.5 Executing Context-aware Hand-Overs

In our hierarchical approach to context-aware hand-over policies, the robot observes the values of context parameters c_1, c_2, and c_3 then samples:

- a hand-over position, g^*, from $\pi(g|c_1)$
- a hand-over trajectory shape, W^*, from $\pi(W|c_2)$
- a hand-over execution speed, τ^*, from $\pi(\tau|c_3)$

The result is then received by the common lower-level policy, represented by $\pi(u_t|x_t;\omega)$, which takes the form of the DMP equations:

$$\tau^*\ddot{\mathsf{y}} = \alpha_{\mathsf{y}}(\beta_{\mathsf{y}}(g^* - \mathsf{y}) - \dot{\mathsf{y}}) + \frac{\Psi^T W^*}{||\Psi||_1}$$

$$\tau^*\dot{\mathsf{x}} = -\alpha_{\mathsf{x}}\mathsf{x}$$

where the forcing term has been written to make the dependence on W^* explicit.

Figure 5.5 depicts the hierarchical policy structure in the form of a probabilistic graphical model.

As a result, the trajectories chosen by the robot would now adhere to all three contextual dimensions, a product of the hierarchical structuring of the learned, contextual policies and the flexible trajectory representation afforded by the DMP lower-level policy. As an example, Figure 5.6 shows four simulated trajectories, each combining certain 'sampled' DMP parameter values (the same ones used to generate Figure 5.4, but in combination).

5.2.6 Contributions

The primary focus of this project was to utilize an existing framework (or multiple frameworks) to achieve context-aware hand-overs, then methodically verify the hypothesized enhancement in a robot's performance of the task. Nevertheless, we extended and improved upon the existing implementation (of C-REPS), as well as developed certain software components that were essential for the HSR to execute reasonably fluent hand-overs. This sub-section outlines some of these notable improvements and contributions.

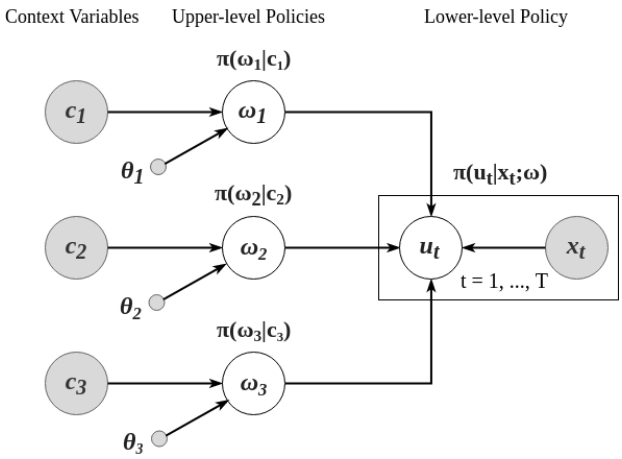


Figure 5.5: Graphical model of the hierarchical policy structure used in our C-REPS implementation

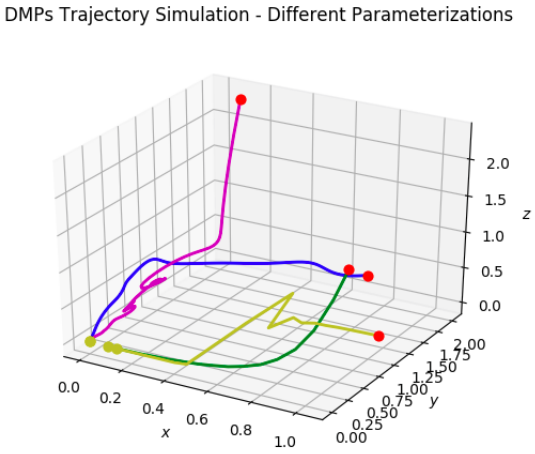


Figure 5.6: 3D trajectories sampled using the results of $\pi(g|c_1)$, $\pi(W|c_2)$, and $\pi(\tau|c_3)$

Incorporating DMPs in C-REPS Implementation The C-REPS Python package developed by Ricardo Dominguez: PyCREPS, on which our model-based C-REPS is based, offers an adequate implementation of the algorithm which retains all essential features described in the original publication ([67]). However, it had originally been written with a particular focus on OpenAI gym environments, and was demonstrated on the benchmark Cart-Pole and Acrobot problems. As this was reflected in the implementation, it was necessary to modify the algorithm to suite our framework.

The upper-level policy was implemented with generality in mind, such that it is independent of the representation of the lower-level policy, which controls the actual action sampling behaviour of the algorithm. For the aforementioned problems, this required simple vector computations and decision rules, for example. Since our work depends on generating trajectories using DMPs, we implement the lower-level policy as a set of DMPs. To that end, we utilize the functionalities of the *pydmps* package, and integrate DMP formalization and subsequent roll-out generation into the C-REPS algorithm.

More importantly, this integration of *pydmps* DMPs is also the basis for our trajectory 'model', which simulates trajectory roll-outs when learning off the robot, and allows us to plot and evaluate executions during the learning process.

In general, PyCREPS was modified and extended to allow systematic plotting and recording of results, among other additions. Note: A lesser modification involved a correction in the upper-level policy covariance matrix update equation, which did not exactly match Eq. (5.12).

Exploration Random Restarts As alluded to in section 5.1, in the discussion concerning the exploration characteristics of REPS algorithms, the problem of premature convergence to sub-optimal parameters was expected. Since this was observed in the results of our learning-by-simulation phase, the algorithm had to be augmented with a measure of regularization, as referred to in the machine learning literature, such that we ensure the learning process is not hindered by instances of convergence on local optima.

The covariance matrix Σ of the upper-level policy, $\pi(\omega|c)$, primarily determines the rate of exploration at each iteration. Occasionally, over-confidence in high reward regions of the search space cause the algorithm to prematurely diminish the values of Σ, bounding the exploration of subsequent updates to within some, possibly sub-optimal region. This was mainly identified as a cause for a deeper problem associated with the upper-level policy $\pi(g|c_1)$: parameter updates were eventually constrained to lie on a line in 3D space, due to the properties of the linear-Gaussian model and the WMLE update equations. This issue is discussed more thoroughly in chapter 6, and section 6.4 in particular.

In order to avoid these problems, we perform random restarts of exploration parameters matrix Σ: every preset number of iterations, elements of Σ are set to random values. This resembles a technique used in random-start hill climbing, a local search algorithm, for the same purpose, and effectively forces exploration parameter values out of potential local optima, restarting the exploration process. Crucially, the values of a and A are retained, such that the algorithm resumes exploration in the same vicinity, but in a less restricted manner. This modification has been found to greatly improve performance, as will be shown in chapter 6.

Person Posture Identification The hand-over skill, implemented using the ROS framework and on the HSR, is enabled by the interaction of multiple software modules, one of which is a person identification component we developed for the project. As described in sub-section 5.2.1, the robot is made capable of autonomously identifying a detected person's posture through a simple heuristic that makes use of bounding boxes provided by an SSD image people detector. The decision of whether a person is standing, seated, or lying down is determined by calculating the ratio between the numbers of pixels that make up the bounding box's height and width:

$$\kappa = \frac{height}{width} \tag{5.19}$$

Subsequently, the decision rule in Eq. (5.14) is used to determine the 'posture' context, with which the robot chooses the most appropriate hand-over position.

Force-based Object Reception Detection Another vital component of the hand-over skill is a module used to detect the forces on the robot's wrist sensor that signify the receiver's reception (or readiness thereof) of the object from its grasp. For this, we treat force sensor readings as a stochastic signal and implement the established CUSUM algorithm for force change detection. Refer to [18] for a gentle introduction on change detection algorithms.

CUSUM is a change detection algorithm, customarily used for fault detection, which detects deviations of a stochastic signal from its nominal mean. This is often regarded as a signal that a fault has occurred in a monitored system; in our work, we utilize this for sensing significant changes in readings of a noisy force sensor.

We assume that force sensor readings z, can be sampled from either of two Gaussian distributions: $p_{\mu_0}(z)$, when the object is held out in the robot's grip with no significant external forces, and $p_{\mu_1}(z)$, when a force resembling a gentle pull of the grasped object is present. At each time-step, the procedure of CUSUM involves computing a cumulative sum of the log-likelihood ratio between the two distributions:

$$S(t) = \sum_{i=1}^{t} s_{ll}(z(i)) = \sum_{i=1}^{t} ln\frac{p_{\mu_1}(z(i))}{p_{\mu_0}(z(i))} \tag{5.20}$$

where s_{ll} denotes the computed log-likelihood.

The decision of whether force sensor readings most likely indicate either distribution is determined by a decision function:

$$g(t) = max(0, g(t-1) + s_{ll}(z(t))) \tag{5.21}$$

If the value of $g(t)$ exceeds some predefined threshold h at any point in time after the robot stops executing a trajectory, the change of distribution from $p_{\mu_0}(z)$ to $p_{\mu_1}(z)$ is sensed, indicating that the receiver is trying to pull the object from the robot's grasp. Parameters μ_1, μ_2, the identical variance of both distributions σ, and h are determined experimentally.

As observed in experiments, and stated by participants of the study, this force change detection scheme results in fluid object reception from the robot, such that the hand-over seems reasonably natural in that specific aspect.

6

Evaluation

Having formalized our problem, developed a hand-over skill representation, and established the procedure we follow using the model-based C-REPS algorithm, we now present initial evaluations of results achieved by the algorithm, including analyses of learning performance and learned policies.

Figure 6.1 shows the trajectory captured in demonstration being simulated by executing a roll-out with the same observed DMP parameters. In our subsequent learning procedure, this shape is altered (in terms of final position and trajectory shape, in particular) as policies that fit multiple contexts are learned through simulations.

Sub-section 5.3.4 contains an explanation of the formalism we use for each aspect of context-dependence relating to our hand-over skill. In this chapter, we describe the practical details of their respective implementations, systematically analyse their performance, discuss various candidate reward functions, and finally elaborate on observed limitations of the algorithm. Sections 6.1 and 6.2 contain these details for the case of learning context-dependent hand-over positions and trajectories, section 6.3 explains why the same formalism is currently inapplicable for the case of execution speeds, and section 6.4 summarizes the observed limitations of the present implementation.

6.1 Learning Context-dependent Hand-Over Positions

In this section, we formalize the process of learning context-dependent hand-over goal positions encapsulated in policy $\pi(g|c_1)$, compare candidate reward functions, and summarize final results achieved in simulations.

6.1.1 Formalization

In order to learn upper-level policy $\pi(g|c_1)$, which determines the best hand-over position given the perceived posture of the receiver, we must initialize policy parameters θ from a demonstration, then define what constitutes a good position for each context, an appropriate reward function, and adequate values for context parameter c_1.

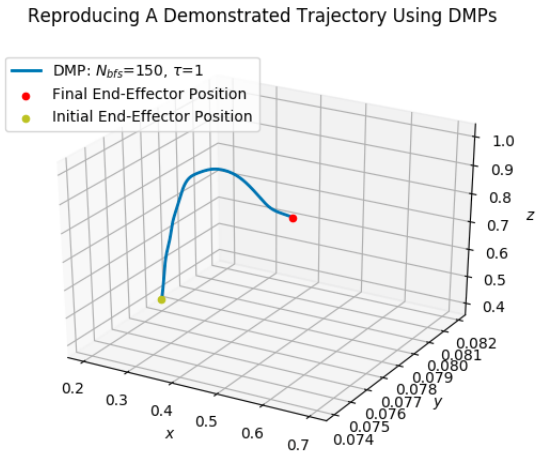


Figure 6.1: Reproduction of a demonstrated trajectory using three DMPs

Considering the nature of the problem, which involves sampling positions in 3D Cartesian space, an intuitive reward signal can be formulated in terms of the distance between a hand-over position sampled from the policy and some 'ideal' position. In addition, we must consider the multi-modality associated with our contextual problem: satisfying every value of c_1 implies having to determine respective ideal positions that define the maximum possible reward for each context.

Reward function $\mathcal{R}_g(g, c_1)$, introduced in sub-section 5.3.4, encodes this logic in a conditional distance-based measure of performance, and is repeated here for convenience:

$$\mathcal{R}_g(g, c_1) = \begin{cases} \frac{1}{||g-\hat{g}^1||_2}, & \text{if } c_1 = \text{standing} \\ \frac{1}{||g-\hat{g}^2||_2}, & \text{if } c_1 = \text{seated} \\ \frac{1}{||g-\hat{g}^3||_2}, & \text{if } c_1 = \text{lying\_down} \end{cases}$$

While we eventually use the Euclidean distance between sampled position g and the respective desired position, as implicated in the equations, we explored similar distance measures, which we discuss in sub-section 6.1.2.

We determine the values of $\hat{g}^1$, $\hat{g}^2$, and $\hat{g}^3$ experimentally. In our lab setting, we assumed the positions shown on Figure 4.2 and moved the HSR's end-effector to different positions while it grasped an object. The coordinates of the positions that felt most adequate for each scenario were then recorded. As a result, the values returned by $\mathcal{R}_g(g, c_1)$ would give the robot an indication of an approximate hand-over region that best suited the receiver's posture. The respective 'ideal' hand-over positions, with respect to the robot's base coordinate frame, were found to be (in meters):

- $\hat{g}^1 = [0.4, 0.078, 1.0]$
- $\hat{g}^2 = [0.5, 0.078, 0.7]$
- $\hat{g}^3 = [0.7, 0.078, 0.7]$

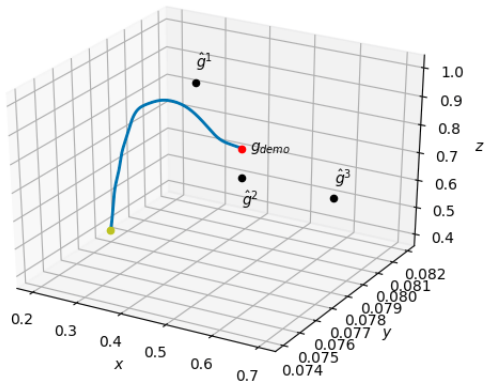


Figure 6.2: Empirically-determined 'ideal' hand-over positions

Note that, while we determined these positions ourselves, a better approach may have been to conduct a brief survey with a number of external subjects whose collective opinions on the best values for each parameter would be used instead.

Figure 6.2 shows the determined 'ideal' positions in relation to the hand-over position extracted from the demonstration, g_{demo}, for the same trajectory shown on Figure 6.1.

Since C-REPS requires a numerical interpretation of our context variables, which we have been referring to symbolically ($C_1 \in \{standing, seated, lying\_down\}$), we ground appropriate values for each. This is a crucial factor in subsequent performance of the algorithm, whose parametric policy representation is heavily influenced by the actual values taken by c_1, as they are a part of the WMLE equations used to update the policy.

Initially, c_1 was chosen to be a simple scalar value, with each element of set C_1 being bound to some arbitrary number that uniquely identified the respective context. Unfortunately, preliminary evaluations of the algorithm's performance showed that, while single-context (learning a single hand-over position) problems were easy, it struggled with the tri-contextual case. In particular, the policy seemed to only be able to learn one position well, and yield poor rewards for others. In addition, the algorithm was observed to converge prematurely on these sub-optimal solutions (the prominent issue of REPS referred to in section 5.2).

This was, in fact, a limitation caused by the linear-Gaussian model of the policy and its update strategy, which placed undesirable constraints that forced strictly linear and diminishing policy updates. In particular, it was observed that sampled positions were quickly constrained to lie on a single line in Cartesian space (as is illustrated in sub-section 6.1.3), which usually passed through the common initial position and close to only one $\hat{g}$.

The plot on Figure 6.3 shows a trajectory simulated to evaluate an initial policy trained for $N = 100$ iterations and $M = 40$ roll-outs per iteration for a problem of learning three chosen hand-over positions

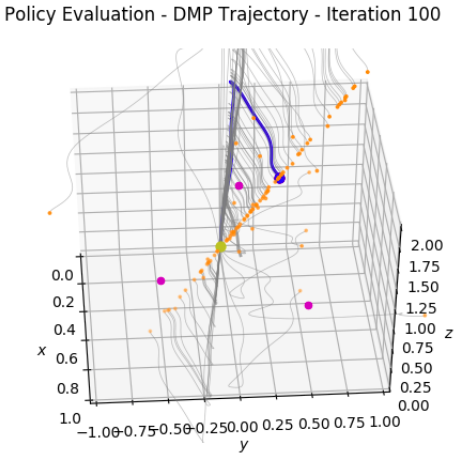


Figure 6.3: Initial policy evaluation, using scalar context variables

(magenta). It demonstrates the problem of $\pi(g|c_1)$ samples collapsing on a single line, which only comes close to one of the points. Here, arbitrary context scalars were used, and the result is convergence on clearly sub-optimal rewards. In this figure (and subsequent ones), the yellow point is the initial position, the blue point is the sampled hand-over position, orange points are positions sampled in earlier iterations, the blue line is the executed trajectory, the thick grey line is the (original) demonstrated trajectory, and thin grey lines represent per-iteration trajectories.

In the following, we describe two measures that were taken to mitigate this problem: one involves a different context variable representation, while the other is an addition we make to the C-REPS algorithm.

Context Vectors and Exploration Random Restarts

The issue of sampled positions being constrained as shown on Figure 6.3 was tackled by using a vectorized representation of c_1 and applying random restarts of the values of exploration parameters Σ. We describe here the method and motivations behind each.

The context variable representation was changed from that of a scalar to vectors that collectively resemble a 1-of-K coding scheme, as illustrated:

$$c_1 = \begin{cases} [x_1] \rightarrow [x_1, 0, 0] \\ [x_2] \rightarrow [0, x_2, 0] \\ [x_3] \rightarrow [0, 0, x_3] \end{cases} \tag{6.1}$$

where x_1, x_2, and x_3 represent the arbitrary values originally selected for c_1. In our final procedure, although random values accomplish the same, we set these values to one of the coordinates of each respective $\hat{g}$.

This has been found to remove the restriction of sampled goal positions lying on a single line, which

now explore along different lines, dictated by the drawn value of c_1. This is illustrated in Figure 6.4a, showing a policy trained with the same conditions as that of Figure 6.3, but utilizing the new context representation. As a result, the policy has a much better chance at approaching the respective desired positions, and exploring within their vicinities. Setting the dimensionality of the context parameter(s), D_c, to 3 transforms A to a 3 matrix, instead of a 1 vector, increasing the number of learnable parameters, which may be the reason for the noticeable improvements, though this must be verified in future work.

Although the policy exploration is thus evidently not constrained to a single line, the samples still exhibit a clear linear correlation, albeit in different regions of the Cartesian search space. This collapse into linear explorations is attributed to premature convergence of some values of Σ, which is addressed by applying **random restarts of exploration**.

A property of maximum likelihood estimations, particularly when used to optimize the parameters of some parametric model, is an underestimation of variances in the underlying distribution. Since they estimate parameters θ directly from the observed data, this strategy is known to suffer from restrictive perspectives of the distribution it attempts to estimate, and over-fitting to the data (in machine learning terms). Our policy update equations are a variant of the same strategy, and thus appear to impose the same restrictions on Σ. Another reason may be the choice of a linear-Gaussian model, whose solutions present a form of linear regression, possibly leading to the observed linear constraints.

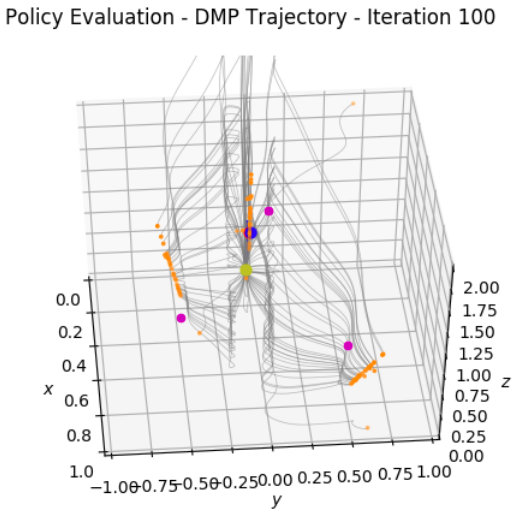
As described in sub-section 5.3.6, covariance matrix Σ of $\pi(g|c_1)$ controls the rate of exploration, and rapidly vanishing elements of the matrix cause the policy to explore only along certain regions or within some sub-space. The WMLE update equations seemingly cause this as a side-effect, leading to occasions where values of the other two parameters a and A are not explored sufficiently. This suggests a simple approach resembling a technique used in random-restart hill climbing, where the values of Σ are regularly re-set randomly by sampling from a uniform distribution:

$$\Sigma \sim \mathcal{U}(0, k) \tag{6.2}$$

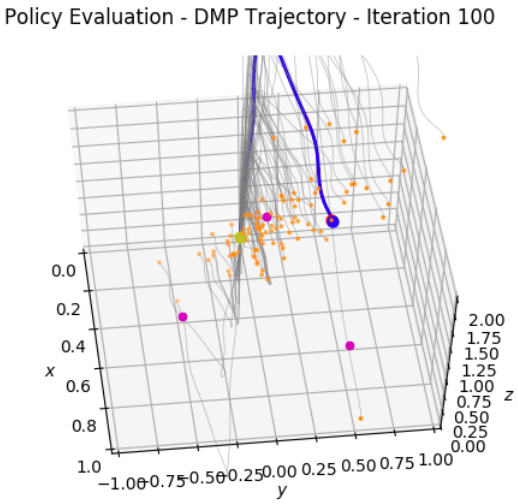
where k is a tunable parameter whose value depends on the problem.

This added step helps the algorithm perform a more balanced exploration, since it resets the values of Σ before the policy converges to some local optimum. Figure 6.4b shows the result of applying random Σ restarts, while still using context scalars. It is evident that updates are now not constrained to one line; instead, the policy escapes the linear constraints and samples goals from around the region. This provides more width to the search, and increases the chances of observing higher rewards and approaching the desired regions, if they are situated laterally to the sub-space initially explored by the policy.

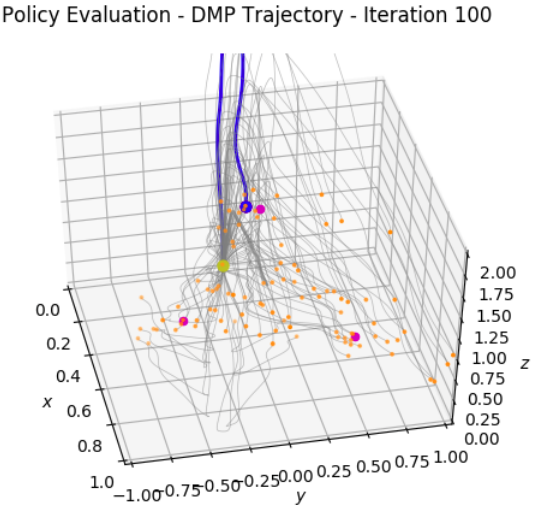
Finally, Figure 6.4c shows the exploration of the algorithm in the same problem after combining the two strategies. It can be observed that the policy samples points that neither lie on a single line nor adhere strictly to a linear form of exploration. We have achieved the best results with these techniques, which allow the algorithm to learn all of $\hat{g}$ in a small number of iterations (less than 200), as well as arbitrary hand-over positions, in policy $\pi(g|c_1)$. Nevertheless, the relative distance between the points directly influences the the number of iterations required to converge to an acceptable solution.



(a) Using context parameter vectors



(b) Using random exploration restarts



(c) Using context parameter vectors and random exploration restarts

Figure 6.4: Policy samples when learning three hand-over positions. The figure shows the exploration characteristics of the policy with a) context vectors instead of scalars, b) random restarts of Σ, and c) a combination of both strategies.

6.1.2 Reward Functions

We have chosen a Euclidean distance-based similarity measure to reward sampled hand-over positions based on closeness to the desired goal positions, for each context. Instant rewards are calculated as the L_2 norm of the difference between g and $\hat{g}$. The resultant reward function, $\mathcal{R}_g$, is relatively simple and has produced the best results. Nevertheless, we explored other similar distance measures as candidate reward functions, and provide a brief description of each in the following.

Manhattan Distance

The Manhattan distance measure simply constitutes the L_1 norm of the differences between two vectors:

$$d_{man}(g, \hat{g}) = ||g - \hat{g}||_1 = \sum_{i=1}^{n} |g_i - \hat{g}_i| \tag{6.3}$$

A much simpler distance measure than the L_2 norm, the Manhattan distance provides less sensitivity to vector differences, which translates to its application in a reward function. As expected, the resultant rewards provide much less differentiation between promising and undesirable states in the policy's search than those of the Euclidean distance measure, a generally unpleasant property in a reward function. This has been observed in the differences in learning speed: the Manhattan distance-based reward function eventually learns similar policies, but at a much slower rate.

Chebyshev Distance

The Chebyshev distance relies on the L_∞ norm, or the supremum norm, to calculate the distance between two vectors as the maximum distance between two respective coordinates:

$$d_{cheb}(g, \hat{g}) = ||g - \hat{g}||_\infty = \max_i(|g_i - \hat{g}_i|) \tag{6.4}$$

This distance measure is more 'pessimistic' than others, since it assigns the highest possible distance across all dimensions between two vectors as the global difference between them. Therefore, when used in a reward function, it is expected to more strictly penalize undesirable sampled positions. While this could help avoid sub-optimal solutions, it may cause the algorithm to overlook promising directions in the search space as it attempts to repel parameter updates from regions that are perceived to be bad along a particular direction. This leads to cases in which the algorithm stifles exploration and thus potentially slows down or impedes learning.

Note The Minkowski distance is a generalization of the Manhattan and Chebyshev distances, alongside Euclidean distance, since it is defined as the L_p norm. As a result, various distance measures can be formulated with different values of p, but the ones considered here are the most common.

Cosine Similarity

Cosine similarity, although not a conventional distance measure, measures the similarity in orientation between two vectors as the cosine of the angle between them:

$$sim_{cos}(g, \hat{g}) = \frac{g \cdot \hat{g}}{||g||||\hat{g}||} \tag{6.5}$$

The cosine similarity presents a less straightforward (and less intuitive) measure of the agreement between our sampled positions and the desired ones in 3D Cartesian space, in comparison to the other measures. Nevertheless, it was put into consideration to study its effects as a reward signal in the policy search. In contrast to the other measures, its values lie in the range $(-1, 1)$, with two vectors yielding a value of 1 if they have the same orientation, 0 if they are orthogonal, and -1 if their orientations are diametrical.

Canberra Distance

The Canberra distance is a version of the Manhattan distance which similarly computes the sum of absolute coordinate differences between vectors, but weights each with the inverse of the sum of the absolute coordinate values:

$$d_{can}(g, \hat{g}) = \sum_{i=1}^{n} \frac{|g_i - \hat{g}_i|}{|g_i| + |\hat{g}_i|} \tag{6.6}$$

It essentially performs a weighted version of the L_1 norm on the vector differences, and is thus expected to yield similar, if not identical, results.

Bray-Curtis Dissimilarity

The Bray-Curtis measure of vector dissimilarity is usually used outside the fields of machine learning and robotics, and can be calculated as:

$$d_{bc}(g, \hat{g}) = \frac{\sum_{i=1}^{n} |g_i - \hat{g}_i|}{\sum_{i=1}^{n} (|g_i| + |\hat{g}_i|)} \tag{6.7}$$

which, again, bears some similarity to the Manhattan distance.

Analysis

In Figure 6.5, we display a comparison between the evolution of reward signals obtained from reward functions based on each of the described distance measures. We attempted to learn a tri-contextual policy for the same problem as that of figures 6.4 and 6.3, for a total number of 500 iterations, then plotted the different average reward signals at each iteration. Note that the actual learning was performed using signals from the Euclidean distance-based $\mathcal{R}_g$, while respective signals from other reward functions candidates were also evaluated per iteration. At each iteration, we average over the rewards the current policy parameters receive for each possible context, yielding the mean rewards.

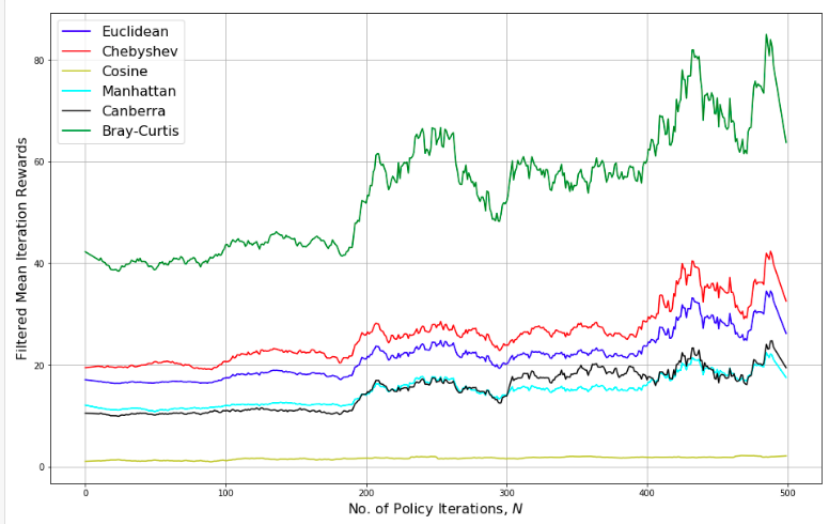


Figure 6.5: Filtered mean iteration rewards vs. number of iterations. A comparison of distance-based reward functions for learning $\pi(g|c_1)$

As expected, the trends exhibited by all of the distance measures are fairly similar, except for the cosine similarity. In addition, the Manhattan and Canberra distances yield nearly identical results, although the latter shows more variation, possibly due to its weighted heuristic. The Euclidean and Chebyshev reward signals are nearly identical, and in fact produce similar results. During learning, the Euclidean and Bray-Curtis distance reward functions were found to perform best on average, learning more stably and having the best results when evaluating the final policy. Nevertheless, the Bray-Curtis measure, when evaluated qualitatively, seemed to produce more 'risk-seeking' policies that were more strongly influenced by sudden high rewards. This occasionally lead to learning one position very well, at the cost of the rest, or achieving sub-optimal rewards for all.

In conclusion, **the Euclidean distance-based reward function was determined to be most appropriate**. Apart from being intuitive and simple, no improvements in performance were observed from the other, less conventional distance measures.

Filtering Reward Signals The iteration rewards plotted on Figure 6.5, and subsequent figures, are filtered using a Savitzky-Golay filter in order to obtain a smoother plot. Since the raw reward signals are relatively erratic, this step ensures reward signal characteristics are more easily distinguishable. The Savitzky-Golay filter computes a moving average of a signal through a convolution-like procedure, fitting every pair (or more) of adjacent points to some polynomial approximation (in our case, a linear one).

6.1.3 Result

The model-based implementation of C-REPS successfully learns an upper-level policy $\pi(g|c_1)$ that starts from he demonstrated behaviour and gradually learns to choose hand-over positions that resemble

the ones implicated in the reward function, depending on current context c_1.

Figure 6.6 displays a solution found by the algorithm for the three 'ideal' hand-over positions defined in sub-section 6.1.1, using the following set of parameters:

- Number of iterations, $N = 1000$
- Number of roll-outs per iteration, $M = 40$
- Relative entropy bound, $\epsilon = 1$
- Number of DMP basis functions, $N_{bfs} = 150$
- Frequency of random restarts: every 200 iterations

Parameters $\theta = \{a, A, \Sigma\}$ were initialized from the demonstration, as previously described. The context vectors were initialized as in Equation (6.1), and their values were:

$$c_1 = \begin{cases} [0.4, 0.0, 0.0], & \text{if standing} \\ [0.0, 0.4, 0.0], & \text{if seated} \\ [0.0, 0.0, 0.7], & \text{if lying down} \end{cases}$$

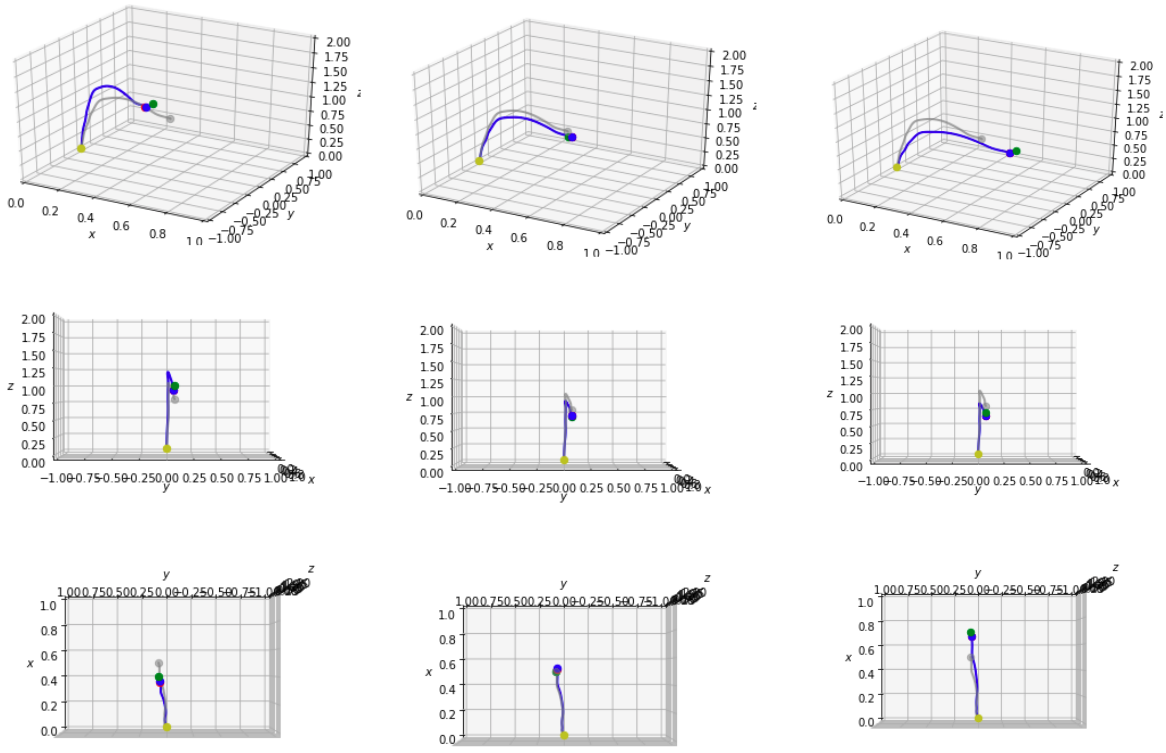
The algorithm ran for **4.33 minutes** on a quad-core XMG laptop with an Intel Core i5-4210M processor and 8GB RAM. The trajectory roll-outs sum up to a total of 40,000 ($N \times M$) simulated hand-overs, which would have clearly required significantly more time to run on the robot directly, in addition to the extreme tedium of the task.

Crucially, the displayed solutions are obtained by sampling the policy with no exploration, by setting $\Sigma = 0$. The exploration parameters only serve the purpose of facilitating the learning procedure, while executing learned behaviour and evaluating its results are done by exclusively sampling positions determined by learned parameters a and A:

$$g \sim \pi(g|c_1) = \mathcal{N}(a + Ac_1, 0)$$

The performance of the algorithm is clearly dictated to an extent by the initializations of Σ, its values at random restarts and the frequency of random restarts. While the algorithm has been able to learn the same goals in far fewer iterations (the positions eventually used on the robot for the study were learned in 160 iterations), longer training times reduce the uncertainty in rewards, as parameters a and A of $\pi(g|c_1)$ stabilize, and lead to less fluctuating samples and thus definitively better final results. Hence, we demonstrate the outcome of training on 1000 iterations here.

Rewards Figure 6.7 displays a plot of the mean (smoothed) iteration rewards vs. the number of policy iterations on the top, and rewards for each context value in the bottom plot. The plot shows that the actual learning process of the algorithm is not entirely predictable, and that a higher number of iterations may not always improve results. This may be attributed to the relatively close distances between the points, such that the algorithm approaches a satisfactory solution early on. Unfortunately, the exploration parameter values do not converge as expected, leading to a subsequent continuation



(a) Standing (b) Seated (c) Lying down

Figure 6.6: Simulated roll-outs with a learned policy, for each respective context. In each figure, the green point signifies the 'ideal' position for that context.

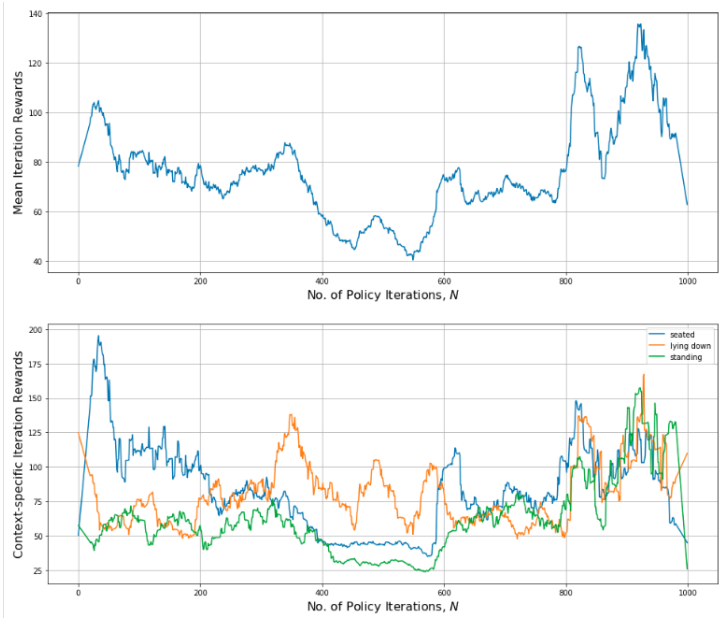


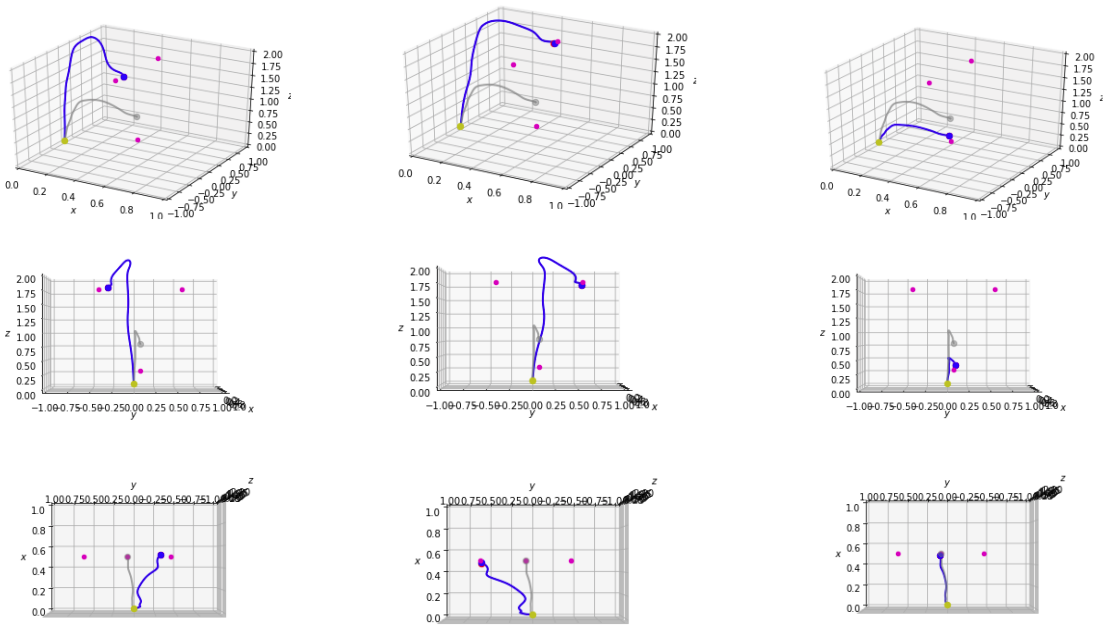
Figure 6.7: Mean iteration rewards (top) and context-specific iteration rewards (bottom)

of exploration, regardless of the present solution. The difficulty faced by the algorithm in achieving a compromise between the rewards received for all contexts can be illustrated in the bottom plot, which shows the variance in reward signals decreasing as they approach similar values. It is worth noting that high rewards for one context causes a decline for the others, revealing a balance that must be struck by the algorithm.

In order to demonstrate the algorithm's ability to handle arbitrarily defined 'ideal' hand-over positions (and their respective reward functions), we display on Figures 6.8, 6.9, and 6.10 the results of learning three sets of different $\hat{g}$:

1. Triangular pattern:
   $\hat{g}^1 = [0.5, -0.4, 1.8]$, $\hat{g}^2 = [0.5, 0.6, 1.8]$, $\hat{g}^3 = [0.5, 0.1, 0.3]$
2. Horizontally collinear points:
   $\hat{g}^1 = [0.5, -0.9, 1.8]$, $\hat{g}^2 = [0.5, 1.1, 1.8]$, $\hat{g}^3 = [0.5, 0.1, 1.8]$
3. Distant, arbitrary points:
   $\hat{g}^1 = [5.0, 0.5, 7.0]$, $\hat{g}^2 = [1.0, 10.0, 2.7]$, $\hat{g}^3 = [10.0, 5.0, 0.0]$

The results show adequate learning and sampling of the approximate desired positions, in spite of a few inaccuracies. These problems were solved using the same parameter settings as before. However, training for the positions on Figure 6.10 required significantly more iterations (3000, against 1000 for the other cases) to achieve acceptable results. While these positions were intentionally placed at artificial and exaggerated distances, it suggests that the relative distance between the points has a direct impact on the amount of training required to capture the desired behaviour in $\pi(g|c_1)$.

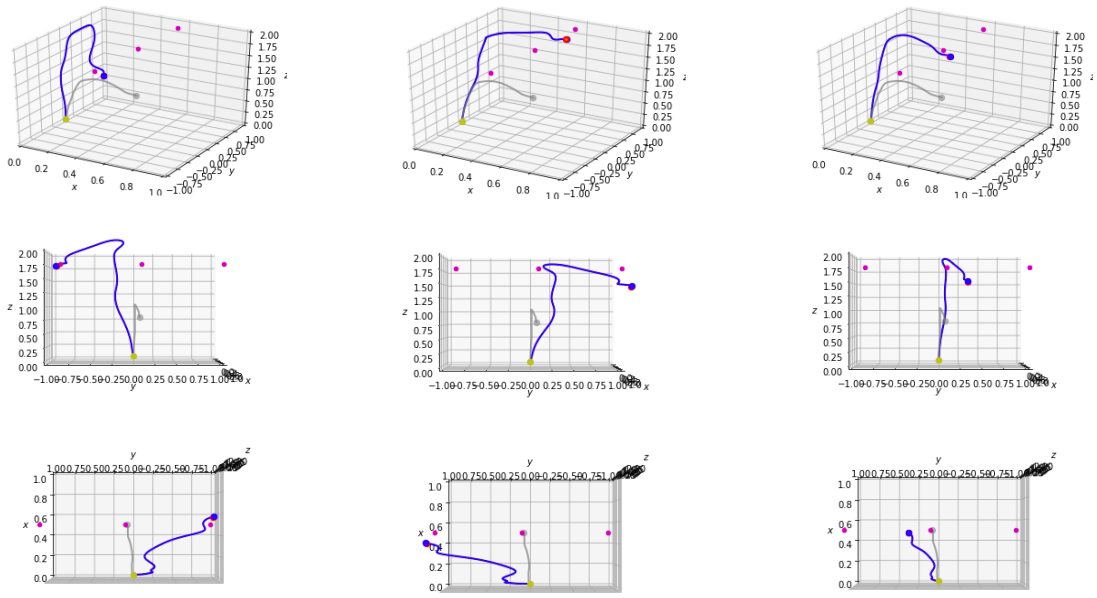


(a) Position 1

(b) Position 2

(c) Position 3

Figure 6.8: Simulated roll-outs with a learned policy, for three positions arranged in a triangular pattern. The magenta points signify the target positions.

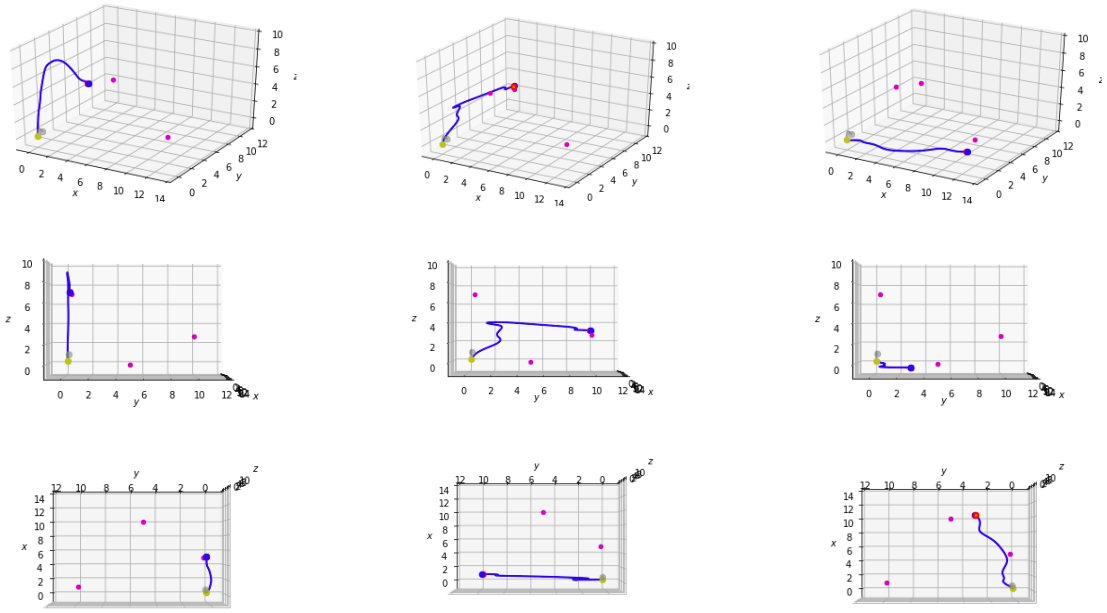


(a) Position 1 (b) Position 2 (c) Position 3

Figure 6.9: Simulated roll-outs with a learned policy, for three positions arranged in a horizontal line. The magenta points signify the target positions.

We conclude this section with a few relevant remarks:

- In the original *PyCREPS* implementation, the policy update equation was implemented inaccurately, such that Eq. (5.12) contained the factor $\omega^{[i]} - a$, instead of $\omega^{[i]} - \mu^{[i]}$. Before correcting this, the algorithm performed poorly, as values of Σ were found to explode during learning.
- We attempted an approach that, instead of drawing a random context value for each roll-out, attempts to fully train the policy on each context in sequence. In other words, policy parameters θ were fully learned while drawing the same context for a given number of iterations, then the learning process was continued with the same θ for the next c_1, and so on. Unfortunately, the algorithm was found to fully converge only on the desired goal of the current target c_1, losing the multi-modality of its solution. This presents a case for regular observations of each value of c_1 being an important factor in learning the contextual policy.
- Initialization of the policy (from the demonstration) can be done using two methods that yield similar final results: setting a to ω_{demo} and A to zeros, or setting a to zeros and formulating matrix A to ensure that the initial $\mu(c) = \omega_{demo}$. However, initializing through a is preferred since it avoids an initially skewed version of the demonstrated trajectory, which is a side effect of initialization through A. This is because the former does not depend on the initial value of c_1, as the latter does.



(a) Position 1 (b) Position 2 (c) Position 3

Figure 6.10: Simulated roll-outs with a learned policy, for three positions arranged arbitrarily at relatively far distances. The magenta points signify the target positions.

Conclusions The algorithm has been shown to perform well in learning position-dependent contextual policies for arbitrary hand-over positions in simulations, under the guidance of reward function $\mathcal{R}_g$. This has been achieved with a relatively small number of iterations in some cases but, as the results show, the final outcome of the algorithm is marginally unpredictable. This is attributed to occasional undesirable updates in Σ parameters, which may disallow completely landing maximal rewards (by settling on the desired positions). Nevertheless, augmenting the algorithm with random restarts of Σ to escape collapses into linear updates, and context vectors to allow exploration on multiple regions of the search space have yielded favourable results.

6.2 Learning Context-dependent Hand-Over Trajectories

In this section, we formalize a process of learning context-dependent hand-over trajectories under policy $\pi(W|c_2)$, compare candidate reward functions, and summarize final results achieved in simulations. We also explain the limitations of our C-REPS algorithm as observed in this case.

6.2.1 Formalization

We take the same steps, as described in chapter 5, for tackling the problem of learning policy $\pi(W|c_2)$ as we do for $\pi(g|c_1)$. This policy must determine the best hand-over trajectory, given by DMP basis function weights W, for the particular scenario; in this case, the presence or absence of an obstacle.

Once again, this requires initializing the parameters of $\pi(W|c_2)$ from the provided demonstration, crafting reward functions incorporating knowledge about 'ideal' trajectory shapes, and assigning numerical values to the binary context parameter c_2.

We follow a similar approach when shaping the algorithm's reward functions. Whereas we simply defined 3D hand-over positions $\hat{g}$ to guide the policy into learning the desired position for each context, doing so with trajectories is more complex. Since parameter W contains the N_{bfs} weight values of the DMPs, certain trajectory shapes were simulated to extract their weight vectors. As a result, we could theoretically employ comparable vector dissimilarity measures in our reward functions, to guide the algorithm into learning multiple such trajectories with $\pi(W|c_2)$, starting from the demonstration trajectory. As an alternative, we could also rely on the raw trajectory coordinate vectors y, as will be seen in the next sub-section.

A Limitation of the Algorithm Through preliminary tests, it was determined that the algorithm is unable to simultaneously learn two (or more) trajectory modes in the same contextual policy $\pi(W|c_2)$. In spite of various parameter settings and reward function formulations (refer to sub-section 6.2.2), all resultant policies were either able to capture one of the trajectories well, or average over the desired trajectory shapes of each context. This may be a result of the sheer number of parameters to be learned in comparison to the former case (as mentioned in the discussion of the comparative complexity in sub-section 5.3.4). This limitation hinders our objective of learning c_2-dependent trajectories. Since the algorithm succeeds in learning single trajectories, albeit at large numbers of iterations, we display the results for these uni-contextual policies here.

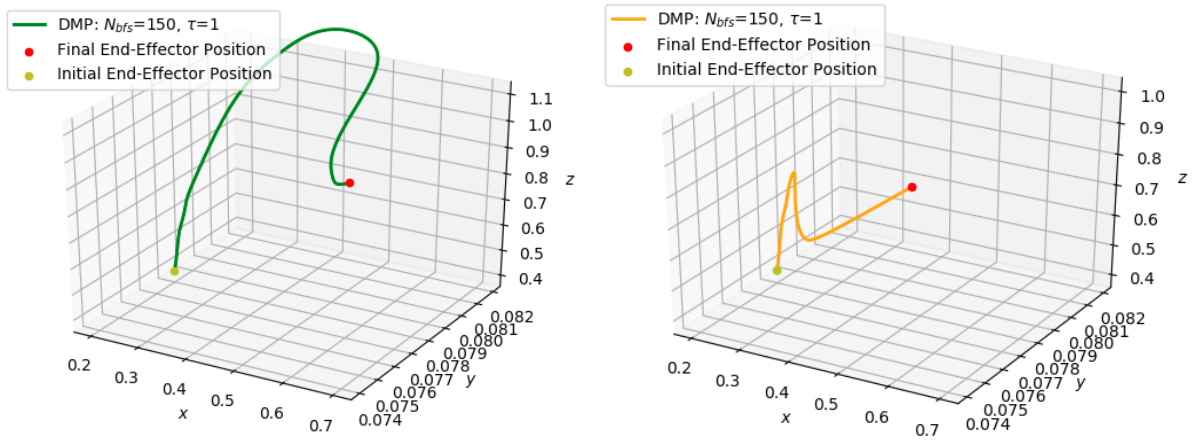
Figure 6.11 shows two example trajectories we artificially shaped as candidates for trajectories to be learned. The first depicts an over-arching motion, while the second resembles the motion of handing an item under some surface. The two trajectories are characterized by weight vectors and trajectory coordinate vectors $\hat{W}^1$ and $\hat{\mathsf{y}}^1$, and $\hat{W}^2$ and $\hat{\mathsf{y}}^2$, respectively.

The parameters θ of $\pi(W|c_2)$ are initialized here in the same manner. Initially a $N_{dmps} \times N_{bfs}$ matrix, W is instead represented as a $1 \times (N_{dmps}\mathrm{x}N_{bfs})$ vector composed of the matrix's stacked columns, to adhere to the algorithm's vectorized update policy equations. Subsequently, a is set equal to demonstration weights W_{demo}, while A is initialized as a square $(N_{dmps}\mathrm{x}N_{bfs})$-dimensional zero matrix where $\Sigma = I$.

We train upper-level policies with a reward function that incorporates some notion of similarity between the desired trajectories and the ones executed as a result of $W \sim \pi(W|c_2)$. The reward function used to generate the final results of the algorithm consists of an inverse of the sum of point-wise trajectory differences (PWTD), in addition to a term enforcing trajectory smoothness:

$$\mathcal{R}_W(\mathsf{y}, c_2) = \begin{cases} \frac{1}{d_{PWTD}(\mathsf{y},\hat{\mathsf{y}^1})} + \eta_{SAL}(\mathsf{y}), & \text{if } c_2 = True \\ \frac{1}{d_{PWTD}(\mathsf{y},\hat{\mathsf{y}^2})} + \eta_{SAL}(\mathsf{y}), & \text{if } c_2 = False \end{cases} \tag{6.8}$$

This function will be described further in sub-section 6.2.2, which elaborates on the PWTD distance measure, the added Spectral Arc Length (SAL) smoothness measure, and other candidates for reward



(a) Over-arching motion trajectory

(b) Low motion trajectory

Figure 6.11: Simulated artificial trajectories, reproduced using DMPs.

function $\mathcal{R}_W(\mathsf{y}, c_2)$.

We also employ **random restarts of exploration**, which have been found to result in better performance, on average. Figure 6.12 shows reward plots for an initial evaluation of the algorithm, which was run once with random Σ restarts every 100 iterations (dash-dotted line), and once without (solid line). Here, the algorithm was used to train a policy for reproducing trajectory $\hat{\mathsf{y}}^2$ ($c_2 = False$) in a low number of iterations (500), to demonstrate the difference (note that the reward characteristics in the trajectory case make observing this effect in reward plots easier than in the position case).

Clearly, the restarts of the exploration parameters force the policy out of performance plateaus, while running the algorithm without the heuristic causes it to prematurely converge to a sub-optimal solution. Note that, while the algorithm initially performs better for the no Σ restarts case: an effect of the stochasticity of the policy, Σ restarts lead the other policy to quickly outperform the former.

This is even more apparent in Figure 6.13 which shows the final trajectory sampled by the trajectory in each respective case. These plots are purely meant to demonstrate the progress of the algorithm for either case, not a final solution. As before, the yellow and blue points refer to the initial and goal positions, respectively, the blue line is the sampled trajectory, the thick grey line is the original, demonstrated trajectory, while the thin grey lines show trajectories sampled in previous iterations to demonstrate the algorithm's progress (note that, due to the slow progression of the policy, samples are close to each other and may collectively appear like thick lines).

Correlated Exploration Strategy An interesting observation was drawn from the comparison of the inclusion and omission of Σ restarts on Figure 6.12. Σ was originally initialized as a diagonal matrix, I, regardless of subsequent restarts, in which the values were sampled from a uniform distribution. The plots revealed that performance was significantly better after the first restart (at 100): the run in which restarts were used (dash-dotted line) initially performed even worse than the run without, but the acquisition of

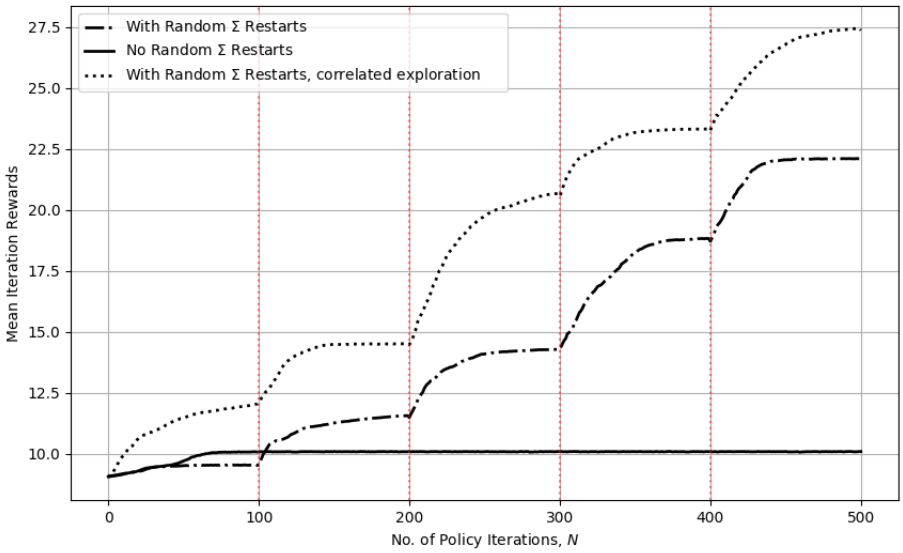
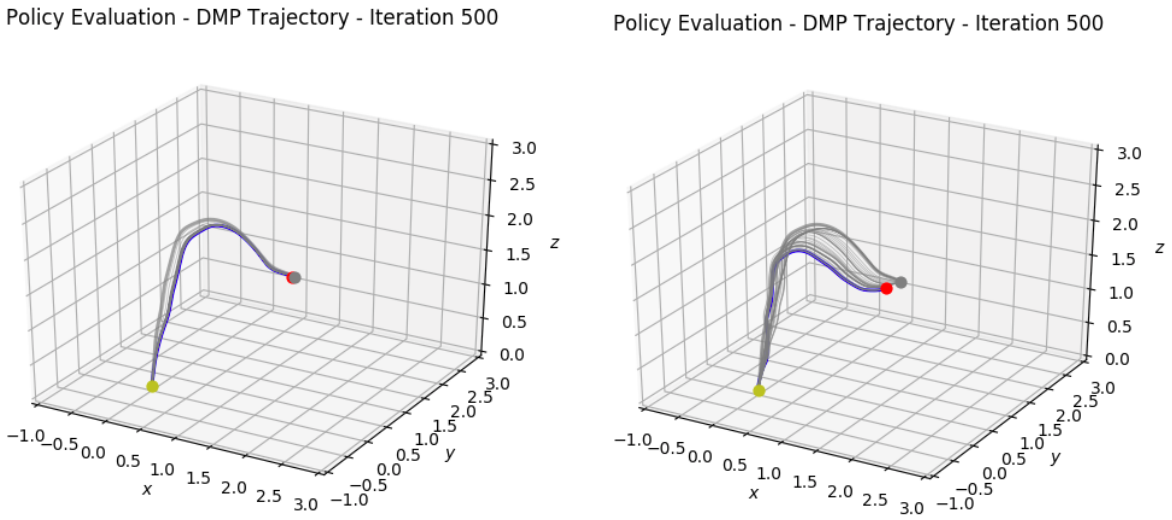


Figure 6.12: Mean iteration rewards vs. number of iterations for the problem of learning $\hat{\mathbf{y}}$ in 500 iterations without Σ restarts (solid) and with Σ restarts: with initial $\Sigma = I$ (dash-dotted) and $\Sigma = \mathcal{U}(0, 2)$ (dotted). The red dotted lines show the iterations at which Σ was restarted for the latter case. The solid and dash-dotted lines correspond to the results in figures 6.13a and 6.13b, respectively.



(a) Policy iteration with no Σ restarts

(b) Policy iteration with Σ restarts every 100 iterations

Figure 6.13: A comparison of the first 500 iterations in the problem of learning $\hat{\mathbf{y}}$. The plots show the progress of the algorithm when a) Σ is unchanged, and b) when Σ values are regularly reset.

higher rewards was vastly increased after the first restart. This suggested that initialization of a full Σ matrix, instead of a diagonal one, would yield better results overall, since the algorithm would start its exploration in a more efficient manner.

This was confirmed in a third run of the algorithm, in which we initialized the full covariance matrix with random values, and also applied random restarts with the same frequency (dotted line on Figure 6.12). A full Σ matrix corresponds to a correlated exploration strategy, which has been shown to vastly increase learning speed. This proves the fact that algorithms like PILCO, which enforces strictly uncorrelated exploration, would perform worse than others like Black-DROPS and REPS algorithms (as conjectured in the comparison of Appendix A). The fully correlated exploration strategy is thus adopted hereafter.

6.2.2 Reward Functions

Similar to the position-learning case, we utilize measures of vector distances to evaluate closeness of sampled trajectories to the desired ones. Although the concept remains the same, the nature of the problem suggests the use of different measures that incorporate either DMP weights or raw trajectory coordinate vectors, or known trajectory smoothness measures. In the following we describe each considered measure, analyse their characteristics from observed results, and present our chosen composite reward function.

Point-wise Trajectory Differences (PWTD)

The PWTD distance measure evaluates the similarity between two trajectories with coordinate vectors y and $\hat{\mathsf{y}}$ by calculating the sum of the squares of their normed point-wise differences. The inverse of this quantity thus provides a reward function that provides higher reward signals the more the trajectory shapes align. Since this places an undesirable dependence on vector sizes, we do not compare all pairs of points lying on the trajectories; instead we define some equally-spaced sub-set of point coordinates, P_s, whose size specifies the number of pairs compared in the function:

$$d_{PWTD} = \sum_{i \in P_s} ||\mathsf{y}_i - \hat{\mathsf{y}}_i||_2^2 \tag{6.9}$$

The resultant reward function is very similar to the Euclidean distance-based function used to learn context-dependent positions, except that we square the normed differences here.

Euclidean Distance Between Weight Vectors

Since the algorithm effectively searches for the ideal values of W, the distance between the current weight vector and some pre-determined vector corresponding to the desired trajectory can alternatively be used to construct a reward function:

$$d_{WD} = ||\mathsf{W} - \hat{\mathsf{W}}||_2 \tag{6.10}$$

This measure is justified by the invariance properties of DMPs, for which similar trajectory shapes and deformations are theoretically expected to be generated using (at least globally) similar sets of weights w_i.

Note that we could apply the same strategy of the d_{PWTD} of choosing particular points along the vector to reduce the computational burden, but this is consciously avoided here due to the delicate nature of the weight vector. In particular, the evolution of the basis function weights w_i produced by the policy during learning has been observed to be highly unpredictable, as compared to trajectory vectors y. This is due to the DMP equations still possibly producing similar trajectories with significantly different values of W, especially as the size of W grows. It is thus expected that selectively sampling points along the weight vector to be used in the comparative reward function would lead to unreliable performance.

Euclidean Distance Between FFT Weight Vectors

This measure is identical to the weight vector distance measure, but compares the Fourier transforms of the vectors instead. We compute the discrete Fourier transforms (DFT) of the weight vectors, using the Fast Fourier Transform (FFT) method, then apply the same formula to calculate the normed difference between them:

$$d_{FWD} = ||\mathcal{F}\{\mathsf{W}\} - \mathcal{F}\{\hat{\mathsf{W}}\}||_2 \tag{6.11}$$

Here, $\mathcal{F}\{x\}$ denotes the Fourier Transform of x.

Weight Vector Correlation

The weight correlation measure of similarity calculates a correlation coefficient for two same-size vectors as their dot product divided by the product of their magnitudes:

$$d_{Wcor} = \frac{\mathsf{W}^T\hat{\mathsf{W}}}{|\mathsf{W}||\hat{\mathsf{W}}|} \tag{6.12}$$

This measure is inspired by the work of Ijspeert et. al. in [51], in which they employ the same technique to compare profiles of trajectories generated using DMPs.

Spectral Arc Length Smoothness Measure

Measures of movement profile smoothness have previously been presented by Balasubramanian et. al. in [10] and [11], and more recently by Gulde et. al. in [41]. The use cases were primarily related to neurology and the study of gait patterns in aging or unhealthy patients, where motion smoothness was used as an indication of either anomalous or normal movements. While targeted towards a different class of problems, these measures could be useful in achieving reward functions that are well-suited to the adjacent robot trajectory learning problem.

One such measure is Spectral Arc Length (SAL), which quantifies trajectory smoothness based on changes in profile curvature, on the basis of the Fourier spectrum of the underlying movements. In [10],

the authors provide a time-invariant version of this quantity measure, whose formula is given by:

$$\eta_{SAL} = -\int_0^{\omega_c} \left[\left(\frac{1}{\omega_c} \right)^2 + \left(\frac{d\tilde{\dot{Y}}(\omega)}{d\omega_c} \right)^2 \right]^{\frac{1}{2}} d\omega \tag{6.13}$$

It is important to note that ω here refers to frequency (in Hz), not to policy parameters ω: the convention we have adopted here. ω_c denotes some predefined frequency band, while $\tilde{\dot{Y}}(\omega)$ is a normalized Fourier magnitude spectrum of velocity profile $\dot{\mathrm{y}}$: $\frac{\dot{Y}(\omega)}{\dot{Y}(0)}$.

From this, we can derive a simple similarity measure by comparing the SAL values of our sampled and desired trajectories:

$$d_{SAL} = |\eta_{SAL}(\dot{\mathrm{y}}) - \eta_{SAL}(\hat{\dot{y}})| \tag{6.14}$$

This measure deals with the velocity profile of the movement, which can be provided by the *pydmps* implementation of DMP roll-outs, instead of position coordinates in the case of d_{PWTD}.

As expected, and verified in preliminary evaluations, this measure of similarity does not distinguish between scaled versions of the same trajectory, since it only takes curvature similarities into account. In theory, this may hinder a policy trained on a reward function incorporating this measure in learning trajectories with little curvature variations.

Dimensionless Jerk Smoothness Measure

Another popular measure of movement smoothness is dimensionless jerk, particularly when classifying healthy or anomalous human limb movements, since the latter are expected to exhibit jerks in the velocity profile. According to [11], dimensionless jerk of profile $\dot{\mathrm{y}}$ can be computed using:

$$DLJ = -\frac{(t_2 - t_1)^5}{\dot{\mathrm{y}}_{peak}^2} \int_{t_1}^{t_2} \left| \frac{d^2\dot{\mathrm{y}}(t)}{dt^2} \right| dt \tag{6.15}$$

where t_1 and t_2 represent the initial and final time-steps (0 and T), respectively, while $\dot{\mathrm{y}}_{peak}$ refers to the peak velocity throughout the movement.

Being very similar to the SAL measure, the DLJ can be utilized to provide a measure of how similar two trajectories are:

$$d_{DLJ} = |DLJ(\dot{\mathrm{y}}) - DLJ(\hat{\dot{y}})| \tag{6.16}$$

In addition, another variant calculates the log dimensionless jerk instead, but is not considered here.

When used as a reward function, this distance measure was found to not distinguish between scaled versions of the same trajectory shape, like d_{SAL} didn't, suggesting it may not be the ideal option.

Reward Function: Combining Distance and Smoothness Measures

The reward function that we use to obtain our final results makes use of the PWTD trajectory difference and the SAL smoothness measures:

$$\mathcal{R}_W(\mathsf{y}, c_2) = \frac{1}{d_{PWTD}(\mathsf{y}, \hat{\mathsf{y}}, |P_s|)} + C\eta_{SAL}(\mathsf{y})$$

The first term is the inverse of the PWTD between y and $\hat{\mathsf{y}}$: the sum of square normed differences between $|P_s|$ equally spaced points along both trajectories. Clearly, this rewards closeness to the desired trajectory shape $\hat{\mathsf{y}}$. The second term is the SAL smoothness of the trajectory, which gets more negative the more jagged trajectory y is. Apart from ensuring a smooth trajectory in general, this measure also accounts for the fact that the PWTD measure asymptotically results in equivalence of the points dictated by P_s only: the smaller $|P_s|$ is the more jerky the trajectory is. η_{SAL} thus acts as a trade-off factor, similar to a regularization parameter, bringing stability to the policy trajectory samples.

The η_{SAL} measure is based on an open-source implementation by the author (Balasubramanian)[1]. Note that it requires specifying a sampling frequency, which we choose to be equal to $|P_s|$.

Analysis

Figure 6.14 depicts a plot of the normalized reward signals obtained from each of the reward functions based on the aforementioned respective distance or smoothness measures. These were collected on the same problem of figures 6.12 and 6.13, after running the algorithm for 500 iterations.

Due to the high variations in reward magnitudes, the reward signals from each reward function have been normalized by subtracting their means, in order to better be able to compare their reward 'landscapes'.

The figure shows that **the reward function combining point-wise trajectory differences (PWTD) and the Spectral Arc Length (SAL) smoothness measure is more promising than other reward functions**, due to its stability and intuitive scaling with the current performance of the algorithm. While a reward function employing only PWTD exhibits a similar reward signal evolution, the addition of the SAL smoothness measure has been found to noticeably reduce trajectory irregularities in curvature, particularly during early stages of the policy search.

Interestingly, the Euclidean distance between the current and desired weight vectors (WD) varies much less than the trajectory coordinate differences measure. The resultant reward function thus assigns very similar reward values throughout, although the trajectory clearly approaches the desired shape, and would thus slow down or even suspend learning. As previously mentioned, the unpredictable nature of the basis function weight vectors and their updates according to $\pi(W|c_2)$ makes their use in a similarity-based reward function somewhat precarious.

The similar weight vector differences measure for which the vectors were transformed using FFT (FWD) yield the same exact results (its line in the plot is made thick to distinguish it from that of the similar reward function).

[1] https://github.com/siva82kb/smoothness

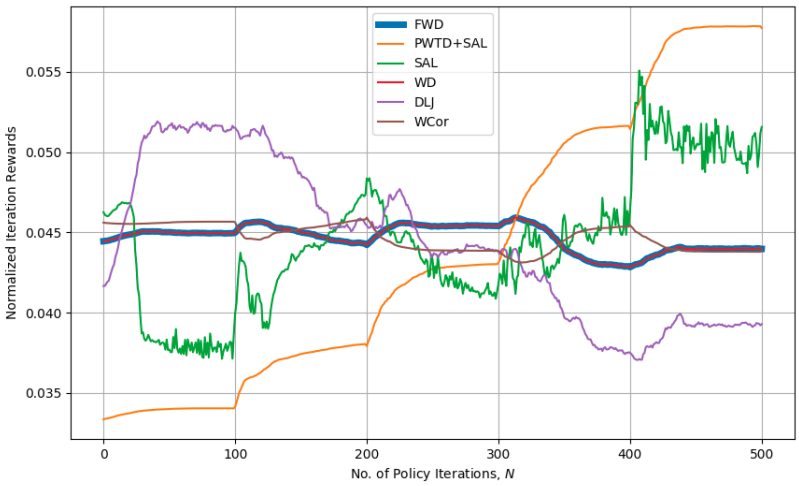


Figure 6.14: Normalized iteration rewards vs. the number of policy iterations. A comparison of the evolution of reward signals from different reward functions for learning $\pi(W|c_2)$)

The pair of smoothness measures, SAL and DLJ, do not produce reliable results on their own, and thus do not qualify as independent reward functions. Due to the focus of the measures on the global curvature characteristics of trajectories, their resultant reward functions are not particularly suited to the problem at hand, as can be seen from Figure 6.14.

6.2.3 Result

In the trajectory learning case, the model-based C-REPS implementation successfully learns separate upper-level policies $\pi(W|c_2)$ that are able to learn and reproduce trajectories similar to the ones implicated through reward function $\mathcal{R}_W(\mathsf{y}, c_2)$. The algorithm was used to learn two policies $\pi(W|u)$ and $\pi(W|l)$, which were trained to reproduce trajectories of Figure 6.11a and Figure 6.11b, respectively.

Figures 6.15 and 6.16 display the progress of the algorithm while learning each policy, under the following set of parameters:

- Number of iterations, $N = 2000$
- Number of roll-outs per iteration, $M = 40$
- Relative entropy bound, $\epsilon = 0.1$
- Number of DMP basis functions, $N_{bfs} = 150$
- Frequency of random restarts: every 100 iterations

Using the same machine, training either policy required about **267 minutes** (accounting for the fact that we used twice as many iterations as for $\pi(g|c_1)$, the algorithm requires about 31 times as much time to train the policy as for the former case). Understandably, the algorithm requires far more computations and, hence, time than the previous case, most likely due to the comparatively large dimensionality of the parameter space.

Note that the parameterizations here are very similar to the position case except for minor adjustments. As expected, the algorithm requires far more iterations than the position policy $\pi(g|c_1)$ to approach desired trajectories. While the position-learning policy could generally achieve acceptable results in less than 300 iterations, learning $\pi(W|c_2)$ seems to require a far larger number of iterations to fully converge to the desired solution. Nevertheless, the algorithm progresses fairly well in 2000 iterations, and particularly in the first 1000, when the current trajectory is farther from the desired.

The algorithm performed better when the relative entropy, ϵ, was set to a lower value of 0.1 (instead of 1). This meant that consecutive policy updates were less bounded and given more freedom to stray away from current behaviour, allowing the algorithm to explore more aggressively in the space of policy parameters. It appears that this is need here more than in the position case, possibly due to the higher number of parameters and thus relative difficulty of the problem.

Figure 6.15 shows evaluations of the policy trajectory samples for the first 1000 iterations in learning the over-arching motion introduced in Figure 6.11a. The policy clearly learns to approach the approximate shape that would yield higher rewards, as its samples progressively tend to increasingly resemble the desired trajectory. It can be observed that the greatest progress happens in the early stages of learning. This makes sense since the magnitude of rewards are significantly lower when the sample trajectory shapes are farther away from the desired shape, increasing the overall value of d_{PWTD}. The more the trajectory approaches that shape the generally higher the rewards per roll-out are, and the more difficult it is for the algorithm to identify the regions of the search space that would yield maximum rewards.

On Figure 6.16, the results are shown in the same form for the case of learning the lower motion trajectory of Figure 6.11b. As with the first case, the algorithm can be seen progressing as it runs for more iterations, sampling trajectories that approach the desired shape. However, the algorithm requires more iterations to progress at a similar rate to the over-arching motion case; in other words, it approaches the desired trajectory at a fairly slower rate. This may be attributed to the same reason that caused the policy updates to grow less impactful late into the learning process for the first case. Since the desired lower motion trajectory is relatively close to the demonstrated (and initial) trajectory, from the perspective of the PWTD similarity measure, the variation in roll-out rewards is essentially less than in the over-arching motion case, leading to a slower learning curve. For this reason, the bottom three plots are shown points that are 500 iterations apart, since differences in the trajectory samples become less apparent, even though the algorithm still continues to make progress.

It is important to note that context parameter c_2 was set to a scalar for this case, since it was less consequential than in the position learning case.

We conclude this section with a few relevant remarks:

- The problem of the algorithm not learning multiple trajectories using the same policy could not be alleviated using the same techniques applied in section 6.1. Particularly, the use of context vectors instead of scalars, did not seem to improve performance as would have been expected: the algorithm still mostly learns a degraded version of either trajectory, or a compromise between the two.

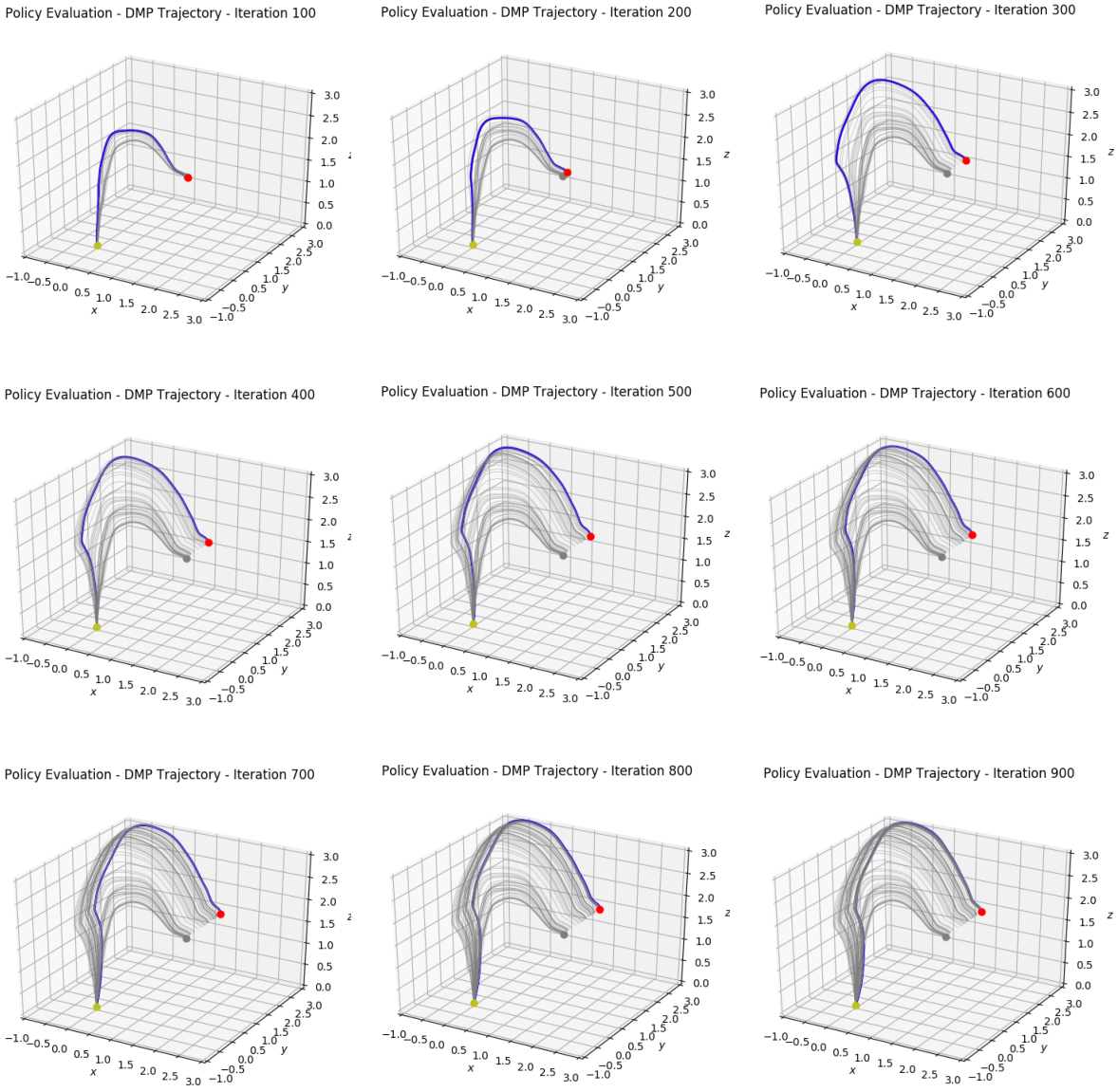


Figure 6.15: The progress of the C-REPS algorithm when learning a policy that starts from the demonstrated trajectory and learns to reproduce the over-arching motion trajectory shown on Figure 6.11a

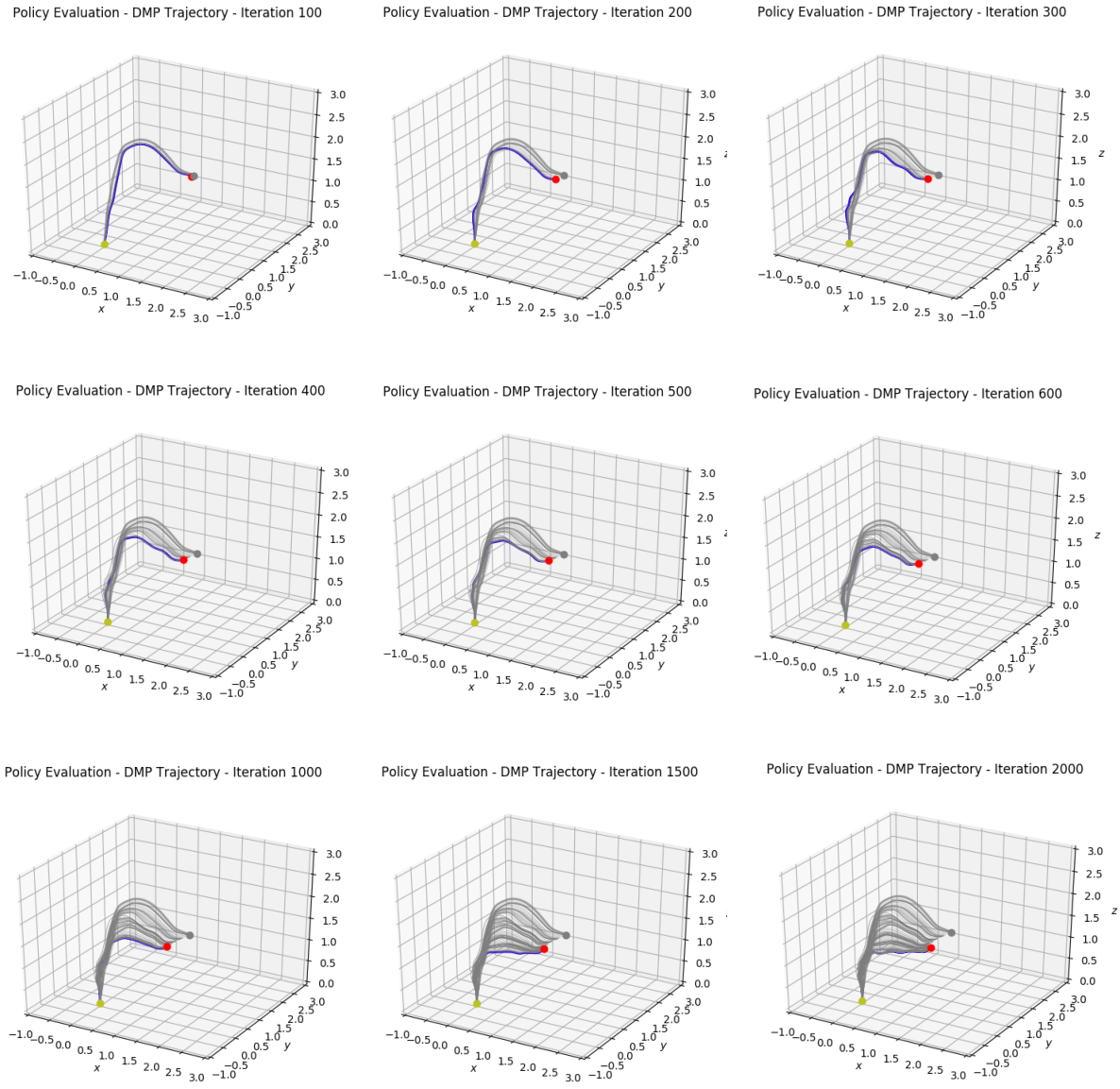


Figure 6.16: The progress of the C-REPS algorithm when learning a policy that starts from the demonstrated trajectory and learns to reproduce the lower motion trajectory shown on Figure 6.11b

- A possible solution for easing the problem, particularly for the multi-contextual case, would be to start at initial W value corresponding to simpler trajectory shapes (such as a straight line). As of now, this approach has not mitigated the aforementioned problem.
- An undesirable product of the rapid changes in DMP basis function weights W has been found to distort the resultant trajectory, such that the achieved final position is not identical to the one designated in the DMP equation. This is not ideal, since learning trajectory shapes would work best for a stable final position. A possible cause for this could be the rapidly growing values of W, which may need to be bounded somehow.
- While reward function $\mathcal{R}_W(\mathsf{y}, c_2)$ works relatively well, it still suffers from an undesirable quality. Namely, the two terms scale differently, since the reciprocal of distance measure d_{PWTD} quickly outgrows the SAL smoothness quantity as the policy converges on the desired trajectory. In the future, an adaptable scaling term would ideally be used to balance the importance of both constituents of the function.

Conclusions From the perspective of learning single complex trajectory shapes solely from a shaped reward function's signals, the C-REPS implementation performs well. Although learning takes significantly longer than in the hand-over position case, the results show that the algorithm steadily improves and approaches the desired trajectory characteristics, particularly with the added random restarts of exploration strategy. The multi-contextual case has proven to be too challenging for the algorithm and/or the current policy representation so far: attempting to learn separate trajectory shapes indicated by respective context parameter values (scalars or vectors) yields unsatisfactory results. At best, the algorithm can currently be used to train multiple $\pi(W|c_2)$ policies, one for each context, and a decision rule could determine which to apply for the observed context. In the future, this issue may be resolved through an improvement of the algorithm, or a change in the policy formulation.

6.3 Learning Context-dependent Hand-Over Speeds

As previously explained in sub-section 5.3.4, the third dimension to the problem of learning context-dependent hand-overs, execution speeds, has not been pursued further in this work. We provide here a brief explanation and some empirical results that demonstrate the inadequacy of the implementation we currently rely on for the aspect of execution trajectory speed.

The third learnable parameter in the DMP equations, τ, allows the time-invariant trajectories represented by the equations to be effectively scaled, such that the speed of their execution is tuned. It is assumed that a trajectory captured through a demonstration, for example, is reproduced at exactly the same speed when executing a roll-out with the corresponding DMPs while setting $\tau = \tau_{demo}$. Theoretically, decreasing or increasing this value leads to roll-outs in which the same motion is executed at slower or faster speeds, respectively.

With this added degree of freedom, as explained in section 4.1, a third dimension of context, for which optimality of task executions are governed by trajectory speeds, can be learned using a third upper-level policy in our hierarchy: $\pi(\tau|c_3)$. As a simple example, we assume c_3 to be a binary variable

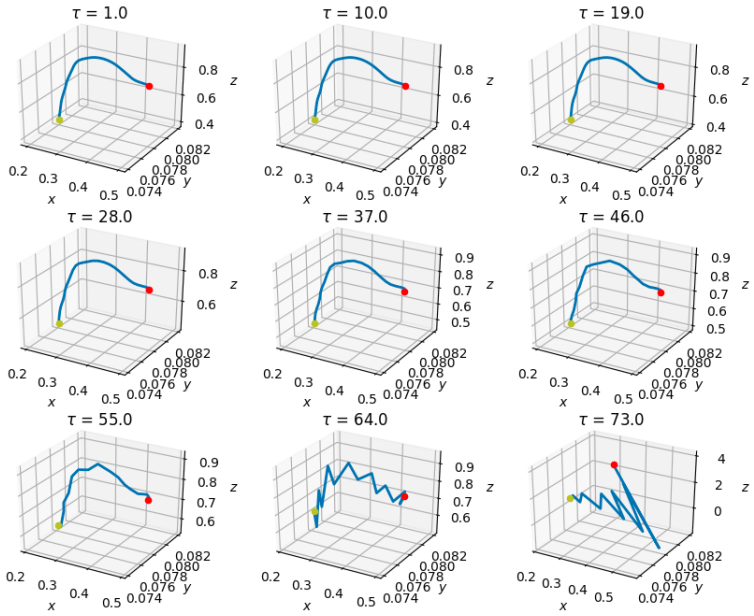


Figure 6.17: A comparison of DMP reproductions of the demonstrated trajectory for different values of time constant τ

denoting whether an object is fragile, and thus necessitates a more cautious movement while handing it over, or not.

Unfortunately, the *pydmps* package, which we utilize for our DMP computations and therefore is instrumental in the implementation of the lower-level policy, has been found to produce undesirable results when tuning parameter τ. This can already be observed on Figure 5.4c. Figure 6.17 shows a comparison of the same (demonstration) trajectory being reproduced for a range of values of τ.

It is evident that the reproduced trajectories grow more jagged and rough as we increase values of τ. While this, from the perspective of the implementation, can be interpreted as a decrease in the time taken to follow the trajectory from start to finish (which qualifies as the 'speed' of the movement), it does not exactly correspond with our understanding of motion execution speed. We expect a slower hand-over to follow the same path, intersecting the same set of points in 3D space, but at a slower pace. These reproduced trajectory roll-outs cause an unnatural discretized movement, as verified on the HSR, which makes the resultant hand-overs inadequate. A more important consequence is that the results of $\pi(\tau|c_3)$ would no longer satisfy the intended conceptual outcomes, which makes learning the contextual policy unjustified.

In conclusion, the context-dependent hand-over speed learning problem was deemed not satisfactorily solvable with the current implementation of model-based C-REPS, and is left for future work.

6.4 Limitations of the Implementation

In this section we briefly summarize the deficiencies and limitations of the proposed approach to learning contextual hand-over policies, as observed from the final results obtained for each aspect of the generalizable hand-over executions. As previously demonstrated, the model-based C-REPS algorithm achieves mixed success overall, learning position-based contextual policies relatively well and attaining acceptable performance on trajectory-based policies. Nevertheless, it is pertinent to enumerate and discuss observed limitations, such that they can be systematically resolved in future work.

Firstly, the algorithm's performance when learning ideal hand-over positions, while ultimately satisfactory, suffers from some unpredictability. In particular, policies often converge to points that do not exactly correspond to the ideal ones dictated by the reward function, and variations in final solutions across different runs is noticeable. While this is expected in an algorithm which is driven by a stochastic policy representation, this behaviour can also be attributed to the inability of the exploration parameters to converge to very low values, signifying certainty in the location of the 'optimal' position. A possible cause may be the form of the reward function, which may need to be altered to inhibit unwarranted exploration. This behaviour was observed both with and without regular Σ restarts.

A significant limitation of the current version of the algorithm is its inability to learn contextual policies for trajectory shapes for problems that require capturing multiple trajectory behaviours in a single policy. Probably due to the sheer number of learnable parameters, as a result of a high number of basis DMP functions being required to reproduce a motion with acceptable accuracy, it is challenging for the algorithm to learn values for the large policy parameter matrices that satisfy multiple solution modes. Future pursuits may include improving the algorithm's policy update strategy, or even changing the policy representation.

As explained in the previous section, our current implementation does not produce the desired results for learning context-dependent hand-over speeds. In particular, the trajectories produced by the *pydmps* implementation of DMPs are inadequate for the intended learning procedure, since they do not produce the expected change in trajectory execution speed. This may be solved by applying a different approach to this aspect of the hand-over motion, but currently prevents learning a meaningful $\pi(\tau|c)$.

While not a literal limitation of the implementation, the context variable definition phase is an apparent concern in the performance of the algorithm. Namely, it is important to define numerical values for these variables that are both conducive to learning the desired mode of behaviour, and have some connection to the underlying concept of context. This is especially difficult when applying the C-REPS algorithm in a scenario where contexts refer to human concepts that govern human interaction subtleties, such as our human-robot collaborative task. Simpler notions of contexts in robotic tasks, such as the location of a thrown ball, are less problematic. Therefore, the task of defining context values that enable achieving desired results adds a dimension to the difficulty of the proposed approach.

Some of the described limitations, and possible unobserved ones, may be ascribed to the policy representation: a linear-Gaussian model. Among the more apparent concerns is the linearity of the model's distribution in the context parameter values c, which allows for far simpler computations, but may place constraints on the ability of the algorithm. One such example is the initial problem of policy updates

being constrained to a line, as described in sub-section 6.1.1 (before enhancing the algorithm with context vectors and random Σ restarts). Even if the adverse effects of this assumption do not clearly manifest in the problem addressed here, they may appear in larger problems, in which the relation between sampled parameters and the conditioned variable is ideally non-linear.

Finally, our model-based approach achieves acceptable results in reasonable time-frames, but does not seem directly applicable on the actual robot system yet. An ideal scenario would involve the robot employing the algorithm model-free: executing hand-overs in real-time and using collected data to continuously update its policies in a *life-long learning* procedure. Additionally, a inverse reinforcement learning (IRL) method could be employed to fully automate the process of reward function shaping and reward acquisition (refer to section 3.1 for a short overview of IRL). However, the number of roll-outs needed to obtain the current results is prohibitively large, and would ideally need to be optimized before the algorithm is ported to the robot. This, however, constitutes one of the long-term goals of autonomous robot skill learning and thus motivates future work.

7

Results

This chapter presents the results of the HSR hand-over user study conducted as a part of this project, and a subsequent analysis to draw relevant conclusions.

The study involved 10 participants taking part in the experimental procedure described in our methodology (refer to section 4.2). The aim of the study was to evaluate the behaviour of a learned contextual policy $\pi(g|c_1)$ which selects appropriate hand-over positions, according to the current context, in comparison to a context-unaware policy: choosing the same hand-over position regardless of person posture, c_1. Subsequently, we would draw conclusions on the merits of applying apprenticeship learning to learn context-aware human-robot hand-overs, and thus the incorporation of contextual knowledge in a collaborative task of this sort.

As previously described, participants receive an object (a water bottle) from the HSR is three different contexts: $c_1 \in \{standing, seated, lying\_down\}$ (see Figure 4.2), under two 'behaviour modes': a context-unaware (B1) and a context-aware one (B2). The sequence of the experiment along either dimension is randomized between participants, who are initially oblivious to the difference between the two modes.

At each run, the robot detects and perceives the person's posture, identifying the current value of c_1. It then executes a hand-over of the object in its gripper using a parameterized DMP policy with a pre-set initial end-effector position, motion trajectory, and time constant. The variable parameter, final hand-over position g, is set to the value observed in an original demonstration (g_{demo}) in mode B1, and sampled from policy $\pi(g|c_1)$, trained using model-based C-REPS, in mode B2. This policy is the one whose results are presented in section 6.1 and displayed on Figure 6.6.

Following the three hand-overs under either behaviour mode, the participants are instructed to grade their agreement, on a 5-point Likert scale, with statements that are designed to estimate their perception of the robot's behaviour. These statements are listed here:

1. The robot performs the action in a way similar to how I would
2. The robot performs the action in a way that best fits the current situation
3. The robot's action feels natural
4. I feel understood by the robot
5. I feel comfortable interacting with the robot
6. I feel safe interacting with the robot

7. It does not take much effort to take the object from the robot
8. I had to strain my arm to receive the object
9. I could predict the intention of the robot

At the end of the experiment, the participants indicate their preference over the two modes on a 5-point Likert scale, and answer a set of general questions. These open-ended questions are intended to gather insights that cannot be obtained from the initial bounded responses, in addition to notable aspects that may have been overlooked in the course of this work. The posed questions are:

- Among the two modes, which behaviour did you find more appropriate/preferable? And why?
- If the robot could learn your preferences, is there anything you would want it to consider when deciding how to hand you an object?
- Do you have any comments or suggestions for improving the robot's behavior? If so, please describe them.

The full questionnaire used in the study can be found in Appendix B.

7.1 Quantitative Analysis

In this section, we present a quantitative analysis of the part of the study in which different aspects of the robot's behaviour were graded by the participants, and their final evaluations of the context-aware behaviour against context-unaware behaviour.

As explained in sub-section 4.2.3, we analyse the results of the Likert-style statements by interpreting the answers as scores in a range of $(-2, 2)$, ranging from strong disagreement to strong agreement (where 0 signifies indifference). Subsequently, we aggregate the scores across participants for each statement, which governs some factor of the robot's behaviour, by computing the sums of the participants' scores. As a result, we obtain an aggregate score in the range $(-20, 20)$, which signifies how positively (or vice-versa) the respective statement is agreed upon, on average. This is done for each behaviour mode.

In tables 7.1 and 7.2, we display the scores and their aggregates over all participants. We summarize relevant conclusions that can be drawn from this data in the following.

The robot's hand-over executions are found to be significantly more similar to human executions, suitable to the current situation, and natural in the case of context-aware behaviour, B2, as opposed to B1. Although the aggregate scores on these aspects for B1 were neutral on average, they were far more positive in the former case. This is in agreement with our main hypotheses, in which we assumed that context-dependent task executions would be perceived by people as more human-like, and generally perceived to be more adaptive to the current situation.

The robot is additionally perceived to be more understanding by the participants when it seems to change its action across the different postures of the person. This similarly aligns with our human intuitions: individuals who seem to actively take into consideration the present conditions of a person, even for a task as simple as handing an object over, are perceived to be more understanding and even considerate. The participants' input suggests the same applies when the giver is a robot.

Although receiving the object from the robot was moderately comfortable for the receivers in either behaviour mode, it was more so in B2. This may be partly due to aforementioned implicit perceptions of

naturalness in motion, and perceived understanding, which induce a psychological feeling of comfort, but could also imply actual physical comfort afforded by the hand-over positions chosen by the robot's policy.

With regards to perceived safety, the results show no difference between the participants' feeling of safety as the robot handed the object over under either behaviour. This is expected since the robot's movements are generally non-intrusive, including the trajectory it executes, and slow. Nevertheless, this also indicates that contextually appropriate behaviour may not change how safe people feel interacting with a robot. This aspect is usually tied more to the physical appearance and other external characteristics of the robot, and less on perceived intelligence.

The participants generally agreed with the statement that suggests they did not need to expend much effort or exert themselves when the robot was under the context-aware behaviour than otherwise. This is closely connected to the amount of strain the participants felt they had to place on their arm to receive the object. In the latter case, a more lucid preference was observed for B2, with the participants implying that they had to strain their arms significantly less on average, even though the general opinion was neutral for the case of B1. Again, this is an intuitive, if not obvious result, since the single position chosen by the robot in B1 is not ideal for at least two contexts, while the context-dependent positions learned and applied by the robot under B2 are close to ones empirically determined to be most appropriate for each context ($\hat{g}$).

Finally, the perceived predictability of the robot's behaviour during a hand-over was not found to improve with context-aware behaviour, similar to perceptions of safety. This measure is a particularly difficult one, since the participants observe brief episodes of the robot's behaviour, which themselves do not portray much of its general behaviour to enable drawing a general conclusion. It is expected that a more involved experimental procedure and a larger number of overall trials, in which participants interact more deeply with the robot, may offer more insightful results on this behavioural factor.

Figure 7.1 provides a useful visualization of the results of this part of the questionnaire in the form of a diverging stacked bar chart. Note that the scores for the eighth statement: "I had to strain my arm to receive the object" have been inverted for the purpose of these plots, such that the scores are a measure of lack of strain on the person's arms, instead of presence thereof. This was to mirror the positive notions of the other statements, and thus clearly demonstrate that the number of positive responses on the bottom plot exceed those of the first: a visual illustration of the participant's general inclination towards the context aware behaviour.

The final evaluations of the robot's behaviour by the participants of the study are presented on Table 7.3. Using the same scheme, the scores from the question of how much B2 is preferred over B1, which is answered on a Likert scale from 1-5 (not at all, to definitely, respectively), are reformulated so that they lie in the range $(-2, 2)$. Therefore, the aggregate score lies in the range $(-20, 20)$, ranging from strong preference for the robot's behaviour under B2 to strong preference for B1 (once again, note that the participants are unaware of what each behaviour mode actually constitutes, at the time of the study).

Out of the ten participants of the study, the scores imply that six strongly preferred the robot's hand-over behaviour under the context-aware mode, B2, two preferred it at a lesser degree, and two preferred behaviour under context-unaware mode, B1. The eventual aggregate score was 12, signifying

Factor	Participant No.										Aggregate
	01	02	03	04	05	06	07	08	09	10	
Similarity to Human Execution	1	-1	-1	1	1	1	0	0	-2	0	0
Suitability to Current Situation	1	-2	0	1	1	2	0	0	-2	0	1
Naturalness	1	0	0	0	2	1	1	-1	-2	1	3
Perceived Understanding	-1	-1	1	1	1	2	1	1	-2	1	4
Comfort	2	0	1	1	1	2	2	-1	-2	1	7
Perceived Safety	2	0	2	2	2	2	2	2	1	1	16
No General Exertion	1	-1	2	1	2	-1	0	1	-1	1	5
Strain on Arm	-1	0	1	-1	0	1	1	-1	2	-2	0
Predictability	1	-1	1	-1	2	1	1	1	1	1	7

Table 7.1: Results of scores of the statements on page 1 of the questionnaire for B1: context un-aware behaviour, over three hand-over scenarios, for participants 01-10. Each statement is scored in the range $(-2, 2)$: from strong disagreement to strong agreement.

Factor	Participant No.										Aggregate
	01	02	03	04	05	06	07	08	09	10	
Similarity to Human Execution	1	1	1	-1	2	2	2	2	-1	1	10
Suitability to Current Situation	1	0	1	0	2	2	2	1	1	1	11
Naturalness	1	0	1	-1	2	2	2	1	0	0	8
Perceived Understanding	-1	1	1	1	1	2	2	0	1	1	9
Comfort	2	1	2	1	2	2	2	1	0	1	14
Perceived Safety	2	0	2	2	2	2	2	1	2	1	16
No General Exertion	1	2	1	1	2	2	1	1	-1	-1	9
Strain on Arm	-1	-1	-1	0	-1	-1	-2	-1	1	-2	-9
Predictability	1	-1	1	-1	2	2	1	1	-1	1	6

Table 7.2: Results of scores of the statements on page 1 of the questionnaire for B2: context aware behaviour, over three hand-over scenarios, for participants 01-10. Each statement is scored in the range $(-2, 2)$: from strong disagreement to strong agreement.

	Participant No.										Aggregate
	01	02	03	04	05	06	07	08	09	10	
Overall Preference (B2/B1)	-1	2	1	-1	2	2	2	1	2	2	12

Table 7.3: Result of scores indicating overall preference of context aware behaviour (B2) over context un-aware behaviour (B1) on a range of $(-2, 2)$ over all trials for participants 01-10.

Figure 7.1: A diverging stacked bar chart used to visualize the results shown on tables 7.1 (top plot) and 7.2 (bottom) for the results of the questionnaire for the context un-aware and context aware behaviours, respectively.

that the context-aware behaviour was generally preferred over the other, as is expected.

7.2 Qualitative Analysis

In this section, we briefly analyse the qualitative results of the robot's perceived behaviour in our study, drawing notable conclusions from participants' vocal inputs and answers to the general questions posed in the questionnaire. Due to the similarity in opinions and notions expressed by the participants, we provide here a general summary of the extracted insights.

The most notable comments provided by the users are summarized in the following:

- The majority of users expressed a strong (vocal and/or written) preference for the results of the context-aware hand-overs, even though they did not know the underlying difference between the two behaviour modes.
- Many participants explicitly mentioned the robot's action literally feeling more natural and more human-like in the context-aware mode, due to them more strongly feeling that it perceived their current posture, during that set of trials. This comment was purely a result of them noticing the robot's adjustment of hand-over positions across contexts, as opposed to the other case.
- Many participants echoed the importance of perceiving as many aspects of their posture as possible, including how far they leaned back on a seat, for example.
- A relatively common remark suggested that straighter, simpler trajectories are preferable, especially since some curvature in the robot's hand-over motion was noticeable, especially in the standing scenario. While this is a product of the reproduced demonstration trajectory, not the output of learned policy $\pi(g|c_1)$, it is an interesting and previously unappreciated view.
- Some users noted that the robot's motion seemed more natural when it moved its full body (arm and base) as it approached them. This was observed only in one case: lying down and under context-aware behaviour, for which the robot was forced to move its whole body to approach the learned farther position. This could imply that the hand-over skill may improve by incorporating whole body movements more often.

With regards to the open-ended questions concerning the how the perceived robot behaviour could be improved, some recurrent suggestions included the following:

- Some users felt the readiness and attention of the receiver should be taken more actively into account by the robot as it prepares to execute a hand-over. Namely, the robot could detect whether a person's attention permits the object transfer, as well as provide more active signalling of when it starts executing the action.
- A few users mentioned more advanced posture detection capabilities as a welcome improvement, such as detection of orientation at which a person is standing and how far they are leaning while seated.
- A similar comment made by one user included suggestions of utilizing an added detection modality, such as hand position detection, such that the most comfortable position can be actively estimated, or even tracked.

It is worth noting that various users were noticeably placing undesirable focus on the force detection component when evaluating the robot's behaviour, even though they were explicitly told that objection reception detection is not the main focus of the study. This may have occasionally distracted users away from observing the hand-over execution itself, leading to less discriminative opinions on the robot's context-dependent hand-over behaviour, as opposed to its counterpart.

Lastly, some variation in the participants' preferences for some postures reveal the difficulty in grounding what could be considered 'good' behaviour in the hand-over task, even for humans. As an example, the seated position had fairly varying viewpoints, with some participants ultimately unsure where they would prefer the hand-over then, and one participant suggesting offering the object at shoulder height: a preference that significantly conflicts with our personal intuitions. It could be the case, however, that having a larger group of participants may ease drawing some stable conclusions on what constitutes the generally preferred behaviour, such that we maximize the soundness of the contextual knowledge we guide the robot's learning with.

8

Conclusions

Future autonomous service robots must satisfy user expectations that extend beyond quantitative task performance criteria, to qualify as more than mere robotic tools. One such expectation is context-awareness, a measure of autonomy that can potentially elevate perceptions of robot intelligence, and equip robots with capabilities to meaningfully assist and collaborate with humans. Contextual adaptivity is a vital human characteristic that has tangible effects on behaviour, particularly in joint activities, and thus the development of similar adaptation strategies in robots designed to interact and collaborate with humans is a beneficial pursuit.

In this project, we explore an approach to acquiring and generalizing a robot motor skill to different situational contexts, then conduct a human user study to evaluate its results; with the aims of demonstrating a concrete application of context-awareness, and validating its merits to justify its implementation. We equip an HSR with a context-aware hand-over skill which is enabled by dynamic movement primitives (DMPs), a flexible trajectory representation, and context-dependent behavioural policies, learned by reinforcement. In particular, we employ an apprenticeship learning procedure: the robot is shown a single demonstration of the task, motion characteristics are captured in DMPs, and a model-based policy search algorithm is used to learn a hierarchical structure of contextual policies which facilitate hand-over executions that adhere to the current context, by selecting appropriate meta-parameters of the movement representation.

The algorithm of choice is a model-based version of Contextual Relative Entropy Policy Search (C-REPS), whose information-theoretic policy update strategy and policy hierarchy present useful properties for addressing the problem of context-dependent hand-overs. Policies that choose optimal hand-over positions and trajectories for factors such as the receiver's current posture are learned through 'mental rehearsal': running hand-over trajectory simulations off the robot and guiding learning through shaped reward functions. The results of the algorithm are evaluated both in simulations and in a dedicated human user study. Participants receive an object from the HSR in different postures (standing, seated, and lying down) under the demonstrated and the learned policies, and then provide their inputs on either behaviour through a questionnaire.

Our model-based variant of C-REPS successfully learns a policy $\pi(g|c_1)$ that chooses the most appropriate hand-over position, given the current context. It is also capable of progressively learning a desired trajectory shape in $\pi(W|c_2)$, but currently fails to capture multiple shapes corresponding to

different contexts simultaneously, in the same policy. We determined that learning context-dependent execution speeds is currently infeasible, due to limitations of the DMP implementation. Through our user study, we confirmed that hand-over positions chosen by learned policy $\pi(g|c_1)$ are generally preferred over a context-independent behaviour along various dimensions. This leads us to accept our hypothesis concerning the improvements brought about by context-aware executions of a human-oriented task, achieved through reinforcement learning (RL).

To conclude this report, we briefly summarize our contributions, learned lessons, and avenues for future work.

8.1 Contributions

The primary achievements of this project are a systematic procedure for acquiring a generalizable robot motor skill through apprenticeship learning, using it to implement context-aware robot-to-human hand-overs, and experimentally proving the superiority of resultant behaviour over a conventional approach, through a user study.

This section summarizes the main contributions of the project.

Review of Approaches to Robot Motor Skill Generalization

In order to identify and differentiate between plausible approaches to the addressed problem, we conduct a review of the state-of-the-art robot motor skill learning and generalization methods, which is presented in chapter 2. We thus present an extensive overview that encompasses:

- the use of contextual knowledge in robotics and adjacent fields, as well as various formalisms for context awareness and contextual adaptivity
- skill representations ranging from adaptive controllers to probabilistic models and dynamical systems, the latter particularly utilized to capture human demonstrations, and how contextual adaptivity can be encoded into these representations
- reinforcement learning approaches that facilitate both developing and generalizing robot motor skills, and are conducive to incorporating contextual knowledge to achieve that generalization

We place a particular emphasis on apprenticeship learning, which evolved from the combination of learning from demonstrations (LfD) and RL, and which we identify as the most promising approach to the process of autonomous skill acquisition, improvement, and generalization to different operating conditions. As a result, we explore hierarchical and contextual policy search methods, presenting past applications and reviewing most prominent works in more detail.

In addition, we review research on human-robot object hand-overs, our representative use case, presenting insights from studies on the nature and characteristics of the task as studied on humans, formulations and considerations from a robotics perspective, and applications of learned and adapted robot hand-overs.

Comparative Analysis of Candidate Algorithms

We present a concise comparative analysis of four candidate model-based policy search algorithms that have been identified as the most suitable and promising solutions to the problem addressed in this project. The state-of-the-art algorithms are studied in detail and their policy formulations, policy update strategies, strengths, and drawbacks are discussed to draw conclusions on their respective properties. The results are presented in Appendix A.

This analysis resulted in selecting C-REPS, a variant of GPREPS and REPS algorithms in general, as the most suitable candidate algorithm.

Implementation of Hand-Over Generalization

The achievement of context-aware hand-overs by the HSR is a product of implementing an adequate reinforcement learning procedure and a holistic object hand-over action.

We present a model-based version of C-REPS, applicable to the problem of learning context-dependent positions and trajectory shapes of DMPs. It is based on a implementation present in the *PyCREPS* package, which was designed for simpler RL benchmark problems. Therefore, the implementation was integrated with the functionalities of the *pydmps* package, such that the algorithm now utilizes DMPs as lower-level policies. This change also allows us to simulate trajectory roll-outs and their outcomes off the robot, resulting in a form of predictive forward model. Other improvements include incorporating demonstrated DMPs as initial policies, generating reward signals from crafted reward functions, and rectifying certain inaccuracies in the code to enable productive learning.

A context-aware hand-over action was also implemented using ROS on the HSR, which encompasses receiver detection, posture identification, hand-over execution, and object reception detection. The perception capabilities of the robot are utilized to detect a person, using an SSD detector, whose posture is then determined through a simple heuristic. The hand-over is executed in a ROS action server, which chooses DMP meta-parameters according to the learned policies, conditioned on context parameter values. In order to maximize the natural flow of the hand-over, we also equip the robot with the ability to sense the receiver's reception of the object by implementing the CUSUM change detection algorithm, which makes the decision of whether to release the object, given the forces exerted on the robot's wrist.

Evaluation of Model-based C-REPS

Preceding evaluations on the HSR, we present analyses of the quantitative and qualitative characteristics of the algorithm, whose performance is initially tested in simulations. We evaluate the quality and resultant behaviour of policies trained on desired hand-over positions and trajectories, discussing adequate parameter settings and observed limitations of the algorithm. Additionally, we provide preliminary definitions and a comparison of candidate reward functions for each case, constructed from appropriate numerical measures (distance, similarity, smoothness, etc.).

We verify that the algorithm is suited to learning policies that choose hand-over positions based on context, by demonstrating policies trained to prefer empirically determined 'ideal' hand-over positions for

three given posture contexts, and arbitrarily selected positions in other examples, with sole guidance from reward signals.

Human User Study

As a means for confirming our hypothesis on the merits of contextually adaptive robot motor skills, we present a study providing justification for learning context-aware behaviour, and interesting insights into users' expected and desired behaviours. The results clearly indicate a significant preference for the context-aware behaviour especially through user perceptions of similarity to human executions, suitability to current situation, and naturalness. Participants' expressed preference for the robot's hand-overs as it followed learned policies lends credence to the fact that contextually adaptive behaviour comes off as instinctively more preferable, particularly since users were not informed of the underlying differences between the two behaviour modes of the study. *Among the most intriguing results of this study is that the robot can learn a behaviour that is substantially preferable to users over a conventional one while only guided by an initial demonstration and a scalar reward signal.*

Analysis of the feedback, comments, and suggestions for desired robot behaviour by participants suggests that the choice of what contextual knowledge to impart to the robot in its learning process may be relatively subjective at this stage. The significant variance in people's preferences indicates that it is best to obtain results from more extensive studies in order to more reliably craft reward functions and guide learning in the future.

8.2 Lessons Learned

In this section, we summarize some of the more novel insights gathered throughout this project.

As evidenced by the reviewed literature, the motor skill learning process can potentially be accomplished through a broad variety of trajectory and policy representations, policy search algorithms, and generally skill acquisition strategies that have not been extensively explored. Successful applications of imitation learning demonstrate the use of alternatives to DMPs, such as Hidden Markov Models, probabilistic movement primitives, 'interaction primitives' ([8]), and invariant trajectory representations, all of which possess interesting properties. The skill acquisition and learning process itself, whether based on a learning by reinforcement strategy or otherwise, can vary significantly according to the aforementioned representations and the problem at hand. This variation thus suggests that future problems must be studied closely to determine the particular approach most likely to succeed in attaining a solution.

The process of implementing our solution and subsequent evaluations of its results have revealed that the performance of our model-based variant of C-REPS is dependent on a few key factors. The most conspicuous of these are the policy representation, reward function, and context parameter definition. Most of these seem to apply to policy search algorithms in general, and are thus important to note.

The nature of the policy and its accompanying update (or 'learning') strategy is an evident prime determinant of the algorithm's performance, its capacities, and its limitations. For example, the probabilistic linear-Gaussian model used for the upper-level policies has been found to be a conveniently compact model with the ability to capture various forms of behaviour, enable efficient sampling of 'actions',

and directly encode exploration. In addition to intuitive conditioning on context parameters, modelled as random variables, and a straightforward maximum likelihood estimation (MLE) approach to policy updates, this policy representation is well-suited to our problem. On the other hand, certain difficulties in learning have revealed some undesirable properties, such as the fact that policy updates were bound by linear constraints, a problem that was solved with vectorized context parameter representations and random exploration restarts. It is thus important to take the policy model and its update strategy carefully into account.

The formulation of the reward function, which is currently shaped by the designer in our approach, similarly has a substantial influence on the success of the procedure. As with any reinforcement learning algorithm, the scalar reward signal is the robot's single indication of its current performance and guide to optimal behaviour. Although we provide some initial notion of expected behaviour through demonstration, the robot's subsequent progress relies on the crafted reward function and how well it communicates the intended behaviour. The difficulty in shaping an appropriate reward landscape was apparent when various formulations either failed to achieve any results or did so inefficiently.

A third deciding factor, which is mostly applicable to contextual policy search algorithms like C-REPS, is the definition of context parameters. As observed through initial difficulties in the position learning case, the particular form of the parameters can be crucial to any progress at all. Grounding the concepts behind a context parameter and defining numerical values that are compatible with the algorithm and enable meaningful learning can be a relatively complex process.

The conducted user study provided various interesting insights, chief among which is a verification of our hypothesis concerning context-aware behaviours. Among these insights is an important observation relating to the considerable variations in participants preferences for a hand-over. These variations, and the occasional ambivalence, highlight the difficulty in predicting what behaviour is 'good', which is a requisite to imparting contextual knowledge to the robot. This complication has often been reiterated in the literature concerning human-robot interaction and collaboration, and thus remains a notable research problem. From our perspective, it may suggest gathering contextual knowledge more extensively, possibly from larger groups and more elaborate studies, before applying them to robot skill learning.

8.3 Future work

We conclude this report by enumerating and elaborating on possible avenues for future work, including improvements to the current approach, significant changes in implementation specifics, or novel additions to the adopted procedure.

Gaussian processes (GP) and Gaussian process regression (GPR) can be utilized in our reinforcement learning procedure in multiple ways, owing to their generality as function approximators. Conceivable applications include mapping directly from context parameters to DMP parameters, as is done in [110], GP policy representations, and use as forward dynamics models, such as is done in [66] with GPREPS and in [34] for PILCO. Despite their regression procedure reportedly being computationally prohibitive in high-dimensional representation, they may prove useful for less demanding problems. GPs could therefore be integrated into C-REPS, either as policies or forward models.

The full potential of dynamic movement primitives has not been explored in this work, and utilizing their various extensions could be the target of future work. An intuitive enhancement would be to use DMPs as dynamic policies, as opposed to the deterministic, static implementation, such that motor commands are executed in real-time, taking the full evolution of the trajectory into consideration. This could increase the adaptivity of the hand-over action by adding a reactive element to trajectories, which incorporates measures like hand-tracking (as suggested by participants of the study) and obstacle avoidance. This, however, would be expected to substantially increase the complexity of the problem.

As previously mentioned, a multitude of trajectory representations could be used in the future as substitutes for DMPs. Probabilistic Movement Primitives (ProMPs), for example, offer an interesting paradigm in which trajectories in the form of probability distributions are used to mix or alternate between primitives to produce arbitrarily complex motions. This also opens the door for an approach that relies on multiple demonstrations of the task, which is somewhat of a requisite for ProMPs to approximate desired behaviour. Gaussian Mixture Models also capture primitives from multiple demonstrations, and have frequently been used for trajectory representation and generalization. The invariant rigid body trajectory representation presented in [112] particularly suits skill generalization due to its favourable invariance properties.

In working towards an autonomous apprenticeship learning procedure, we could apply inverse reinforcement learning (IRL) techniques to infer reward functions from demonstrations. This would enable the robot to learn possibly complex behaviour optimality criteria from these demonstrations, or from autonomous executions, albeit in a black-box manner and possibly requiring large repetitions of either. As explained in section 6.4, the current prohibitive number of roll-outs required to learn desired behaviour makes it difficult to apply this approach, even if an appropriate function approximation is achieved. Nevertheless, IRL is expected to play an important role in robot life-long learning, and moving forward from manual reward shaping and/or reward signal assignment.

Another possible direction to explore in the future, which relates to the previous discussion concerning policy representations and updates strategies, would be to explore non-parametric alternatives to C-REPS. The basic REPS algorithms rely on parametric models, such as the linear-Gaussian model we use in our work, and their probability density estimation strategies (such as MLE). In recent work presented in [3] by Abdolmaleki et. al. and [111] by Van Hoof et. al., the local CECER and NP-REPS algorithms have been introduced, both of which employ non-parametric models and techniques, which are thought to alleviate the limitations of the former. Additionally, the RBF-REPS algorithm was presented in [4] for non-linear generalization in contextual policy search, which may reduce issues with linear constraints encountered in the present implementation.

Lastly, curriculum learning can be explored in the future to ease the problem of learning particularly difficult behaviours, or increase learning efficiency in general. These approaches seek to construct or follow a learning schedule, whether that constitutes problem scenarios, drawn task contexts, or problem-specific parameterizations, that maximizes learning speed. The general method is akin to how humans formulate curricula, with easier or more manageable tasks learned first, followed by progressively more difficult ones. This fits naturally in a reinforcement learning problem such as our own, where it may be easier

to learn policies that start from states of higher rewards (that are closer to the goal, for example) in earlier episodes, and to increase the complexity of the problem as the policy starts to grasp the intended behaviour, which could reduce the amount of exploration needed to start acquiring reasonable rewards and kick-start learning. A similar approach used in reinforcement learning, called Hindsight Experience Replay (HER) could also be employed to incorporate objectively 'failed' runs in the learning loop.

A

Candidate Algorithms

This appendix contains a brief review of a group of relevant algorithms that were identified as candidates for application to our problem, their individual properties, and a verdict on each. These model-based policy search problems were chosen since they enable learning in simulation, thanks to their predictive models. This would allows us to utilize a *mental rehearsal* approach to learning the hand-over skill: simulating roll-outs and learning by reinforcement off the robot, then transferring the learned behaviours over and executing the resultant action on the robot. As previously mentioned, this is more data-efficient, safer, and less tedious, albeit at the price of possible model errors and potential bias issues.

The algorithms we review in the following are:

- Probabilistic Inference for Learning Control (PILCO)
- Black-box Data-efficient Robot Policy Search (Black-DROPS)
- Gaussian Process Relative Entropy Policy Search (GPREPS)
- Model-Based Guided Policy Search (M-GPS)

A.1 Probabilistic Inference for Learning Control (PILCO)

PILCO is a gradient-based, model-based policy search (PS) algorithm which was introduced by Deisenroth et. al. in [33] for more data-efficient application of reinforcement learning to complex control robot manipulation tasks. It learns control policies by first learning a dynamics model of the system in question, then employing the model to infer state trajectories and thus analytically estimate expected long-term rewards, which are subsequently used to drive policy improvements. The algorithm's suitability for robotic problems, and continuous action domains in general, is ascribable to its efficient probabilistic modelling and inference strategies. As is the case with most policy search algorithms, as opposed to value-based ones, the algorithm is restricted to episodic tasks. Algorithm 1 outlines the main steps of PILCO. The pseudocode is obtained from [34], by the same authors, which contains an overview of PILCO that is more comprehensive than the one provided here.

Note the use of $\bar{J}_\theta$ instead of J_θ as the policy performance measure, which is to signify PILCO's optimization of *expected long-term cost*, as opposed to *expected long-term return*. Although the two notions are virtually equivalent, we explicitly distinguish cost $\bar{J}_\theta$ from return J_θ (our adopted notation), for semantic clarity. As before, θ refers to the current parameters of policy π.

Algorithm 1 PILCO

init: Sample controller parameters $\theta \sim \mathcal{N}(0, I)$. Apply random control signals and record data.
repeat
    Learn probabilistic (GP) dynamics model using all data
    repeat
        Approximate inference for policy evaluation; get $\bar{J}_\theta$
        Gradient-based policy improvement; get $d\bar{J}_\theta/d\theta$
        Update parameters (e.g., CG or L-BFGS)
    until convergence; **return** θ^*
    Set $\pi^* \leftarrow \pi(\theta^*)$
    Apply π^* to system (execute roll-out) and record data
until task learned

The stated data-efficiency of PILCO: the main reason it is a state-of-the-art RL algorithm in robotics, is the result of learning forward models that capture system dynamics and enable 'off-line' trajectory predictions. Since deterministic predictive models (such as simple maximum likelihood models) introduce errors that hinder their application, due to unrealistic assumptions of their accuracy with respect to the real system, PILCO utilizes non-parametric probabilistic models that express levels of uncertainty in the model: Gaussian Processes (GPs). GPs constitute a (possibly infinite) group of random variables, any subset of which is jointly-Gaussian, producing a distribution over functions that directly encodes uncertainty about the form of the function to be approximated. The model is trained on state-action pairs as inputs, and results in a posterior GP which approximates the observed transition dynamics, and with which successive state predictions can be made, using Gaussian Process Regression (GPR). Its non-parametric natures results in a manifestation of closed-form Bayesian model averaging, in which infinitely many possible dynamics models are averaged for predictions, mitigating the adverse effects of erroneous confidence in a single model ([34]).

As shown on the pseudocode of the algorithm, GP priors are first obtained by sampling random control signals, recording all the resultant data, then training a GP model on those to initially estimate the system's transition dynamics.

In its main loop, the algorithm first performs a policy evaluation step by predicting system trajectories using a deterministic inference procedure, called moment matching, then evaluating $\bar{J}_\theta$. Long-term trajectories predictions are obtained by cascading single, next-state predictions using the GP model. Due to the unwieldy nature of the predictive distribution of the change in state (Δ) for non-linear transition dynamics functions, it is approximated by a Gaussian via moment matching: a unimodal approximation method that avoids the intractability of exact approximations. Subsequently, expected cost $\bar{J}_\theta$ is evaluated from the predicted/simulated trajectories.

PILCO improves the policy at each iteration by searching for the parameters θ that minimize $\bar{J}_\theta$, relying on analytically computed gradients of the cost function, through repeated applications of the chain rule. Analytical policy gradients ([88]) are a common alternative to sampling-based gradients, due to less sampling variances and better scalability with the dimensionality of θ. Using the gradients, a non-convex optimization method, such as CG (conjugate gradients) or BFGS (Broyden-Fletcher-Goldfarb-Shanno),

can be used to efficiently determine optimal parameters θ^* in an iterative procedure. Once the algorithm converges, the policy is updated in order to gather more system data to update the learned forward models. This process goes on until desired behaviour is learned.

Conclusions PILCO is a state-of-the-art model-based PS algorithm, whose properties better enable application to the complex tasks of robotics: a significant challenge for contemporary RL algorithms. Incorporating forward model uncertainty in policy iterations alleviates intolerable model biases, while analytic gradients facilitate conventional unconstrained optimization. In theory, these characteristics make PILCO suitable for learning complex, non-linear policies similar to those required by behaviours in the present work. Notably, the algorithm has been shown to learn significantly faster than contemporary algorithms for a real cart-pole problem and a low-cost manipulator, when learning from scratch (with no prior demonstrations), proving its superior performance when no expert knowledge is provided.

On the other hand, the algorithm relies on an assumption that expected cost is differentiable with respect to the policy parameters, and that the gradient can be computed analytically: which is not guaranteed to be the case. In moment matching, the Gaussian approximation may possibly introduce some bias in favour of computational tractability. For these two reasons, reward function and policy representations may be restricted ([25], [67], [24]). When possible, gradient computation may also be more mathematically demanding than sampling-based approaches, due to cascaded applications of the chain-rule. The optimization procedure presents the danger of getting stuck in local optima in areas where analytical gradients diminish, stunting learning. Moreover, GPs make as few assumptions as possible in unexplored regions, meaning learned forward models may only be confident in areas at which training data was observed, limiting their usefulness in unobserved areas. When it comes to our case, PILCO's capability for learning hierarchical or contextual policies is also yet to be proven.

A.2 Black-box Data-efficient Robot Policy Search (Black-DROPS)

The Black-DROPS algorithm performs gradient-free, model-based policy search, and was introduced as an improvement over other model-based methods that require analytical gradients, such as PILCO. Similarly aiming to minimize robot system interaction time for more data-efficiency, Black-DROPS learns GP forward dynamical models, in addition to GP models of the reward function. Instead of using moment matching for approximate inference of trajectories, it utilizes a Monte Carlo approximation approach to estimate, $\mathcal{G}_\theta$: a 'noisy' version of expected long-term reward, J_θ. It then employs a popular black-box optimizer: CMA-ES, which uses an evolutionary strategy to maximize this quantity, and thus obtain the optimal policy parameters. For a more thorough treatment of the Black-DROPS algorithm, refer to Chatzilygeroudis et. al.'s [25]. Algorithm 2 briefly summarizes the underlying procedure.

Similar to PILCO, the procedure of the Black-DROPS algorithm begins with executing roll-outs directly on the robot, so as to record real-system state-transitions and observed rewards. The number of these roll-outs, which are complete episodes of the task, can be varied. Subsequently, GPs are used to learn both a dynamics model (as in PILCO) and an immediate reward model: an addition that accounts for unknown reward functions. The predictive models are then systematically used to predict trajectories and their outcomes for policy evaluation.

Algorithm 2 Black-DROPS

init: Initialize policy parameters θ
repeat for random number of episodes N_R;
    Execute roll-out using π
    Record state-transitions D and observed rewards R
repeat
    Learn GP dynamics models using data in D
    Learn GP reward model using data in R
    Evaluate policy using Monte Carlo approximation; to get $\mathcal{G}_\theta$
    Update parameters θ; by optimizing $\mathcal{G}_\theta$ using BIPOP-CMA-ES; to get θ^*
    Set $\pi^* \leftarrow \pi(\theta^*)$
    Execute roll-out using π
    Record state-transitions D and observed rewards R
until task learned

While the same cascaded one-step predictions of the next state (and, in this case, reward) are used to finally obtain J_θ, Black-DROPS avoids the computation of analytical gradients and subsequent gradient-based optimization in favour of an alternative approach. Trajectory predictions are performed using Monte Carlo estimation, where the outcomes of different candidate policy parameterizations are assumed to be sampled from a noisy function of J_θ: $\mathcal{G}_\theta$. Optimizing this quantity implicitly maximizes J_θ without explicit computation or estimation thereof, and is accomplished by the noisy function black-box optimizer CMA-ES (which stands for Covariance Matrix Adaptation Evolution Strategy). The rank-based evolutionary algorithm simplifies the optimization process by only ranking sampled parameters by observed performance, instead of pursuing an accurate approximation of J_θ and subsequently deriving them. A significant advantage provided by this optimizer is its ability to exploit multiple cores for computations, possibly leading to vast improvements in efficiency.

Instead of using the basic CMA-ES algorithm, the authors of Black-DROPS utilize an augmented version, referred to as BIPOP-CMA-ES, which provides multiple enhancements, including exploration restarts that mitigate local optima problems, and more intelligent parameter population variance settings. The latter enable, for example, favouring regions of the search space for which there is more certainty.

The optimal parameters provided by the algorithm, θ^* are then used to execute a single, data-gathering roll-out, before the process is repeated. With the new state transitions and immediate rewards, the learned GP dynamics and reward models are updated, and the training loop continues until some desired performance measure is satisfied.

Conclusions Black-DROPS is a candidate model-based algorithm that differs from PILCO in its sampling-based approach to trajectory predictions and possesses a few more desirable properties. It does not share PILCO's restrictions on the representable classes of policies and reward functions, due to the latter's computational constraints (as discussed in 5.1.1). This means that these functions need not be differentiable, for example, allowing the application of various policy representations including neural networks and general GPs. It is also worth noting that learning reward models in Black-DROPS

removes assumptions about knowledge of the reward function of a task. With regards to the approximate trajectory inference step, Monte Carlo roll-outs are more computationally efficient and less error-prone than moment matching. In addition, Black-DROPS's evolutionary optimization strategy constitutes a more global search than that of PILCO, suffering less from local optima difficulties. The combination of these two enhancements also allows parallelization and computation using multiple cores, helping the algorithm outperform similar model-based algorithms. In [24], Black-DROPS and its extensions have been shown to obtain better results than PILCO and other algorithms, on a simulated and a real robot problem.

Exploration in Black-DROPS has an advantage over that of the PILCO algorithm. The latter employs a simplified uncorrelated exploration strategy, where diagonal covariance matrices are used in its predictive model. On the other hand, Black-DROPS achieves correlated exploration through its stochastic optimization method, CMA-ES, by updating the full covariance matrix ([35]), which increases learning speed.

In general, the use of GP models to learn system dynamics and subsequently simulating trajectories makes Black-DROPS susceptible to potential problems similar to those of PILCO. The implicit exploration encoded by GPs thanks to flexible uncertainty bounds may adversely affect data-efficiency: since the GPs are only confident in regions of the state space that have been previously visited, the states it predicts in unseen areas may not agree with the actual system dynamics. Additionally, GPR, which is required for inference, is known to add significant computational complexity to an algorithm, which may not scale well to higher dimensional parameter vectors. Another fact that Black-DROPS shares with PILCO is that it has not been combined with pre-structured policies learned from demonstrations, or used with hierarchical or contextual policy representations, to the best of our knowledge.

A.3 Gaussian Process Relative Entropy Policy Search (GPREPS)

GPREPS is a model-based extension to the gradient-free, information-theoretic Relative Entropy Policy Search (REPS) algorithm, which also aims for data-efficient robot task learning by reinforcement, but with a particular focus on contextual policies. These are trained to enable generalization over multiple known operating conditions of a task using a hierarchical policy decomposition strategy. Apart from this extra dimension to the problem, GPREPS also learns GP forward dynamics models from roll-out data collected on a robot, then simulates trajectories that are used to evaluate its traditionally parametric policy. Policy updates are performed by solving a dual function of a constrained optimization problem, yielding artificial roll-out sample weights which are in turn used to compute a weighted maximum likelihood estimate of the optimal policy parameters. Kupcsik et. al. present GPREPS in [67] and provide a more thorough mathematical formulation in [66]. Algorithm 3 provides pseudocode of GPREPS, in a condensed form.

The contextual variant of the REPS algorithm, C-REPS, is the basis of GPREPS, which simply adds an ability to simulate trajectories, by learning predictive GP forward models. REPS is a gradient-free algorithm that relies on information-theoretic policy updates: it aims to maximize expected reward at each iteration while bounding the maximal policy parameter step size so as to minimize relative entropy,

i.e. information loss between updates. Although model-free, the algorithm can be used for on or off-policy learning, works well with function approximators and kernelization, and is conducive to contextual learning. Deisenroth et. al. recommend REPS as the choice model-free policy update algorithm in their survey of PS methods, [35]. C-REPS achieves context-dependent behaviours by learning an upper-level policy $\pi(\omega|c)$ that chooses lower-level policy parameters ω, given an observed context c, and executes actions using a deterministic lower-level policy $\pi(u_t|\omega, x_t)$. In this original formalization ([67]), states are denoted by x and actions by u, since they are essentially robot control signals. Therefore, by learning an upper-level policy: a linear-Gaussian model, that maximizes expected reward over a distribution of contexts, the algorithm learns parametrized lower-level policies that can generalize over contexts of a task thanks to a hierarchical policy structure.

Note: Although Kupcsik at. al. denote context parameters by s, we instead adopt the notation c, since the former is reserved for state variables in conventional RL literature (as presented in chapter 3).

Algorithm 3 GPREPS

init: Initialize contextual policy ($\pi(\omega|c)$) parameters θ, relative entropy bound ϵ, number of policy updates K
repeat for K policy updates;
    repeat for number of episodes, N
        Observe context c
        Sample policy parameters $\omega \sim \pi(\omega|c)$, and execute roll-out using π
        Record contexts S, parameters Ω, and observed rewards R
    Learn GP dynamics models using all data
    repeat for number of episodes (samples), M
        Sample context c
        Sample policy parameters $\omega \sim \pi(\omega|c)$, and predict L trajectories
        Evaluate policy by averaging over L expected returns; get J_ω
        Record contexts S, parameters Ω, and expected returns J in dataset $\mathcal{D}$
    Update parameters θ:
        Solve constrained optimization problem; get Lagrangian parameters η and $\widetilde{\theta}$
        Compute sample weightings $P = \{p^1, ..., p^M\}$
        Perform weighted MLE using D and P to update policy parameters θ

GPREPS is initialized with the upper-level policy's parameters, which can be random or derived from pre-structured policies, a finite number of policy updates K, and a relative entropy bound ϵ. In the first data collection step, context-dependent roll-outs are executed on the robot to gather data for training GP forward models, in addition to estimating a context distribution for later sampling. A predefined total of N roll-outs are performed on the robot. The trained GP models in GPREPS are distinct in that they also incorporate context parameters c.

Similar to other model-based algorithms, the learned models are used to simulate artificial system trajectories for policy evaluations. In contrast to the PILCO algorithm, and with more similarity to Black-DROPS, trajectory predictions are obtained by sampling from GPs, instead of approximate inferencing methods. In GPREPS, M such trajectories are sampled, where $M >> N$, involving drawing a context variable, sampling upper-level policy for parameters ω, executing trajectories using the lower-level policy,

then computing expected return. In this manner, an artificial data-set consisting of respective drawn contexts, sampled parameters, and estimated returns is constructed as a substitute for data collected directly on the system.

It is worth noting that GPREPS addresses a problem latent in the original model-free REPS, which computed expected returns using single roll-outs introducing undesirable bias to the policy updates and producing what is termed risk-seeking policies ([35], [66]). Having access to predictive forward models, GPREPS uses multiple (L) trajectories to compute an average of expected rewards and mitigate this effect.

The information-theoretic policy updates of REPS and GPREPS require a relatively involved procedure. The problem of finding optimal parameters θ is framed as a constrained optimization problem, where the aim is to maximize average expected reward, maintain a valid joint distribution over context variables and policy parameters (ω), and bound the KL-divergence between subsequent policy updates. This is achieved by minimizing a dual function of a Lagrangian in order to obtain Lagrangian parameters η and $\widetilde{\theta}$ (the latter was changed from the original notation to distinguish it from policy parameters θ). These are then used to assign importance weights, p, to samples according to their estimated returns. Subsequently, a weighted maximum likelihood estimate, θ^*, is used to update the upper-level policy, and the procedure is repeated K times.

Conclusions GPREPS is among the most promising state-of-the-art model-based algorithms, particularly for robotics motion tasks, because of its data-efficiency and contextual adaptivity. Its ability to contextualize behaviour by learning hierarchical policies is an advantage not directly offered by PILCO nor Black-DROPS, which makes it more promising for our use case. Although GPREPS does not seem to place restrictions on reward and policy classes (as PILCO does), its performance was exclusively demonstrated for linear-Gaussian upper-level policies, whose formulation conveniently facilitates initializing policies from demonstrations. A vital characteristic of REPS and its variants is the bounding of entropy between policy updates, which both mitigates unpredictable or 'aggressive' policy changes during learning, and helps in preserving demonstrated policy properties while improving it through a stable learning procedure. Furthermore, DMPs work well as deterministic, lower-level policies for GPREPS, and C-REPS in general, and have been used most often ([67], [66], [12]).

The GPREPS algorithm has provably performed well for simulated problems and a real robot tennis application in [66], but only using linear-Gaussian upper-level policies. The chosen parametric model works well in the algorithm's learning procedure, but it is unclear how more complex policy classes can be trained using the current weighted MLE approach. The applicability of formulations such a neural network policies thus remains unproven.

Overall, the properties of the algorithm make it particularly adequate for our work, which is why it heavily influences the solution we propose in this report.

A.4 Model-Based Guided Policy Search (M-GPS)

The M-GPS algorithm follows a more unorthodox approach to model-based PS; it relies on concepts like information-theoretic policy updates that are similar to those of GPREPS, but employs

significantly different policy formulations. As presented by Levine et. al. in [69] and [70], M-GPS combines trajectory optimization and guided policy search (GPS) to learn parameterized policies for complex tasks with unknown dynamics. To that end, they employ time-varying linear-Gaussian controllers to encode distributions over trajectories, which simultaneously work as a substitute for DMPs and constitute system dynamics models. Multiple such controllers are then used to learn a non-linear policy that matches each and is able to generalize over their trajectory distributions, in a supervised fashion. This method allows training neural network policies, a central feature of M-GPS, allowing more expressiveness, access to a wider class of behaviours, and to better handle partially observable tasks with unknown dynamics.

We provide here a brief overview of M-GPS; the reader is directed to the two aforementioned publications for more comprehensive details. Algorithm 4 (from [70]) shows the main steps in the procedure.

Algorithm 4 M-GPS

1: **init:** Initialize policy parameters θ.
2: **repeat** for K policy updates
3: Execute roll-outs, τ_i, for each linear-Gaussian controller, p_i
4: Update policy parameters:
5: Minimize sum of KL-divergences between policy and each p_i; get θ
6: Optimize trajectories using LQG-like method, and update p_i
7: Optimize/increment dual variables λ_i, to enforce constraints in problem

In a primary trajectory optimization step, roll-outs τ_i are executed on the robot/system to optimize a set of time-varying linear-Gaussian controllers, also referred to more plainly as as trajectory distributions p_i, such that they capture the unknown dynamics of some task. Essentially, each of these locally optimal controllers represents a Gaussian 'policy' corresponding to some initial state distribution (such as task conditions, initial starting point, etc., depending on τ_i). The optimization is achieved using a variant of the Iterative Linear-Gaussian Regulator (iLQG) method, in which each controller is updated in a dynamic programming procedure involving computation of a value function and subsequent parameter updates. Similar to the constrained optimization solutions used in REPS, M-GPS bounds these updates through KL-divergences between subsequent trajectory distributions.

M-GPS combines trajectory optimization with GPS to train arbitrary parameterized policies that generalize over the optimized trajectory distributions, thus creating a complex policy that can handle generalization better than the individual linear-Gaussian controllers. In the iterative procedure illustrated in Algorithm 4, a neural network policy, parameterized by θ, is trained with supervision to match the roll-out samples from each trajectory distribution, while the controller generating these are re-optimized such that they match the current policy. In the original publication, this is presented as a constrained optimization problem, in Lagrangian form, which is optimized to determine parameters θ and p_i, and dual variables λ_i (in steps 5, 6, and 7 of the algorithm). Additionally, the algorithm again makes use of KL-divergences to bound the differences between the learned policy and the controller dynamics of each p_i, between updates.

The complex procedure results in efficient learning of a high-dimensional and complex policy that

incorporates all dynamics represented by the linear-Gaussian controllers, while generalizing to operating conditions not encountered in their respective roll-outs. The success of the algorithm in partially observable and contact-rich manipulation tasks has been demonstrated in [70], and earlier for simpler simulated benchmark problems in [69] .

Conclusions Clearly, the M-GPS algorithm offers distinct properties not characteristic of other model-based PS methods, chief among which is the ability to learn neural network policies. These policy representations are significantly more general and expressive than simpler probabilistic models, but dimensionality and non-linearity issues usually challenge PS methods. PILCO is, in general, unable to learn neural network policies due to differentiability constraints and high computational overhead of GP models. REPS algorithms, while in theory able to handle such policy representations, have only been demonstrated for simpler ones requiring low-dimensional search spaces. An interesting point about M-GPS is the use of linear-Gaussian controllers for trajectory representation, instead of the more common DMPs. Consequently, the procedure avoids reliance on an additional lower-level controller that tracks the target states dictated by the DMP (such as end-effector coordinates, in our case). Since stabilizing neural network policies is difficult, system interaction only through these time-varying controllers makes the algorithm more stable, and avoids having to execute actions from the parameterized policy directly on the robot. Overall, since the algorithm had been shown to handle non-linear, and even discontinuous, task dynamics well, it may seem promising for a case such as our hand-over skill.

Among the side effects of the complexity in the procedure of M-GPS is imposing assumptions about the task and environment of the robot/system, primarily due to Gaussian approximations to trajectory distributions as a result of the choice of controllers [25]. This bears mentioning, despite it being a more general problem affecting a wide range of PS algorithms. In spite of the aforementioned advantages of the algorithm's trajectory representations, they are not able to encode as much information about trajectories as DMPs can. This property specifically makes DMPs the choice approach in our work. As with PILCO and Black-DROPS, M-GPS does not currently appear to be compatible with hierarchical and/or contextual policy formulations, neither has its compatibility with LfD been demonstrated. Both of these facts suggest that the otherwise impressive M-GPS algorithm may not be the best choice for the problem addressed in the present work.

B

Study Questionnaire

P. No.: _____

Hochschule Bonn-Rhein-Sieg

HSR Object Hand-Over User Study Questionnaire

1. Experiment Description

- This experiment involves the HSR handing an object in its gripper over to you.
- The hand-over is performed once in three different scenarios: when you are standing up, seated, and lying down on the couch.
- Each hand-over is performed by the robot in one of two 'behaviour' modes.
- The experiment thus involves six trials of the object hand-over action.
- The procedure is divided into two phases. In the first phase, the robot will randomly be set to one of two behavioural modes, and a single hand-over trial in each of the three scenarios will be performed, in random order. In the second phase, the robot will be set to the other mode, and a single hand-over trial in each of the three scenarios will again be performed, in random order.
- All of the questions included in this questionnaire are designed to estimate your perception of the robot's performance, including how appropriate its motions are and how comfortable it is for you to receive the object.
- In order to evaluate the robot's behaviour in each case, you will have to answer an identical set of questions with a 5-point scale ranging from **strongly agree** to **strongly disagree**, following each phase (page 2). In addition, you will be required to indicate a preference over the robot's two behavioural modes on a 5-point scale, and answer three general questions to conclude the experiment (page 3).

2. Instructions

- Please follow the instructions provided by the person conducting the experiment carefully.
- Please do not touch the robot throughout the experiment.
- Once provided with this questionnaire, briefly go through the questions which you will be expected to answer.
- At the start of each trial, please assume the position instructed to you by the experimenter. Make sure to stay in the designated position throughout the trial.
- The robot will attempt to detect you, then present the object to you.
- Please wait until the robot stops moving, then gently pull the object from its gripper. Once the robot senses your signal, it will release the object, allowing you to take it, before moving its arm back to a neutral position. This concludes the trial.
- Carefully observe the robot's behavior during its execution of the action, paying particular attention to its choice of hand-over position (the final position of its end-effector as it waits for you to take the object), and how it relates to the current scenario.
- In the event that the robot's person detection module fails, please be patient while the trial is repeated.
- At the end of the first phase (the first three trials), please fill out the questions in section 3.
- At the end of the second phase (the last three trials), please fill out the questions in section 4.
- Finally, fill out the final part of the questionnaire on page 3.

Please cross out (X) your choices for the statements in sections 3 and 4.

3. Phase I Questions

B.M.:____

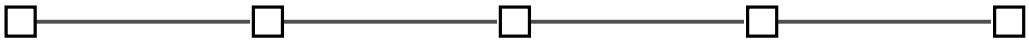
Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The robot performs the action in a way similar to how I would	○	○	○	○	○
The robot performs the action in a way that best fits the current situation	○	○	○	○	○
The robot's action feels natural	○	○	○	○	○
I feel understood by the robot	○	○	○	○	○
I feel comfortable interacting with the robot	○	○	○	○	○
I feel safe interacting with the robot	○	○	○	○	○
It does not take much effort to take the object from the robot	○	○	○	○	○
I had to strain my arm to receive the object	○	○	○	○	○
I could predict the intention of the robot	○	○	○	○	○

4. Phase II Questions

B.M.:____

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The robot performs the action in a way similar to how I would	○	○	○	○	○
The robot performs the action in a way that best fits the current situation	○	○	○	○	○
The robot's action feels natural	○	○	○	○	○
I feel understood by the robot	○	○	○	○	○
I feel comfortable interacting with the robot	○	○	○	○	○
I feel safe interacting with the robot	○	○	○	○	○
It does not take much effort to take the object from the robot	○	○	○	○	○
I had to strain my arm to receive the object	○	○	○	○	○
I could predict the intention of the robot	○	○	○	○	○

<u>Verdict</u>:
Overall, please indicate how much better you found the robot's behavior to be during one phase of the experiment, compared to the other, on the scale below. (Checking the left-most box means a strong preference for the first behaviour, while the right-most box indicates strong preference for the second; the middle box indicates indifference)



I strongly preferred the first behaviour (P1) | **I sensed no difference** | **I strongly preferred the second behaviour (P2)**

5. General Questions

2.1 Among the two modes, which behavior did you find more appropriate/preferable? And why?

2.2 If the robot could learn your preferences, is there anything you would want it to consider when deciding how to hand you an object?

2.3 Do you have any comments or suggestions for improving the robot's behavior? If so, please describe them.

Thank you!

References

[1] Abbas Abdolmaleki, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 154–159. IEEE, 2015.

[2] Abbas Abdolmaleki, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Contextual stochastic search. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 29–30. ACM, 2016.

[3] Abbas Abdolmaleki, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Non-parametric contextual stochastic search. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2643–2648. IEEE, 2016.

[4] Abbas Abdolmaleki, Nuno Lau, Luis Paulo Reis, Jan Peters, and Gerhard Neumann. Contextual policy search for linear and nonlinear generalization of a humanoid walking controller. *Journal of Intelligent & Robotic Systems*, 83(3-4):393–408, 2016.

[5] Firas Abi-Farraj, Takayuki Osa, Nicoló Pedemonte Jan Peters, Gerhard Neumann, and Paolo Robuffo Giordano. A learning-based shared control architecture for interactive task execution. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 329–335. IEEE, 2017.

[6] Muneeb Ahmad, Omar Mubin, and Joanne Orlando. A systematic review of adaptivity in human-robot interaction. *Multimodal Technologies and Interaction*, 1(3):14, 2017.

[7] Jacopo Aleotti, Vincenzo Micelli, and Stefano Caselli. Comfortable robot to human object hand-over. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 771–776. IEEE, 2012.

[8] Heni Ben Amor, Gerhard Neumann, Sanket Kamthe, Oliver Kroemer, and Jan Peters. Interaction primitives for human-robot cooperation tasks. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 2831–2837. IEEE, 2014.

[9] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[10] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, and Etienne Burdet. A robust and sensitive metric for quantifying movement smoothness. *IEEE transactions on biomedical engineering*, 59(8):2126–2136, 2011.

[11] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, Agnes Roby-Brami, and Etienne Burdet. On the analysis of movement smoothness. *Journal of neuroengineering and rehabilitation*, 12(1):112, 2015.

[12] Victor Barbaros, Herke van Hoof, Abbas Abdolmaleki, and David Megerl. Eager and memory-based non-parametric stochastic search methods for learning control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[13] Richard Bellman. *Dynamic Programming*. Courier Corporatio, 1956.

[14] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

[15] Aaron Bestick, Ravi Pandya, Ruzena Bajcsy, and Anca D Dragan. Learning human ergonomic preferences for handovers. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[16] Aaron M Bestick, Samuel A Burden, Giorgia Willits, Nikhil Naikal, S Shankar Sastry, and Ruzena Bajcsy. Personalized kinematics for human-robot collaborative manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1037–1044. IEEE, 2015.

[17] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. *Springer handbook of robotics*, pages 1371–1394, 2008.

[18] Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and J Schröder. *Diagnosis and fault-tolerant control*, volume 2. Springer, 2006.

[19] Domenico D Bloisi, Daniele Nardi, Francesco Riccio, and Francesco Trapani. Context in robotics and information fusion. In *Context-Enhanced Information Fusion*, pages 675–699. Springer, 2016.

[20] Maya Cakmak, Siddhartha S Srinivasa, Min Kyung Lee, Jodi Forlizzi, and Sara Kiesler. Human preferences for robot-human hand-over configurations. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1986–1993. IEEE, 2011.

[21] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[22] Daniele Calisi, Luca Iocchi, Daniele Nardi, Carlo Matteo Scalzo, and Vittorio Amos Ziparo. Context-based design of robotic systems. *Robotics and Autonomous Systems*, 56(11):992–1003, 2008.

[23] Wesley P Chan, Matthew KXJ Pan, Elizabeth A Croft, and Masayuki Inaba. Characterization of handover orientations used by humans for efficient robot to human handovers. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6. IEEE, 2015.

[24] Konstantinos Chatzilygeroudis and Jean-Baptiste Mourer. Using parameterized black-box priors to scale up model-based policy search for robotics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[25] Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58. IEEE, 2017.

[26] Sungjoon Choi, Kyungjae Lee, and Songhwai Oh. Robust learning from demonstration using leveraged gaussian processes and sparse-constrained optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 470–475. IEEE, 2016.

[27] Adrià Colomé and Carme Torras. Dimensionality reduction in learning gaussian mixture models of movement primitives for contextualized action selection and adaptation. *IEEE Robotics and Automation Letters*, 3(4):3922–3929, 2018.

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

[29] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281, 2012.

[30] Christian Daniel, Gerhard Neumann, and Jan Peters. Learning concurrent motor skills in versatile solution spaces. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3591–3597. IEEE, 2012.

[31] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *The Journal of Machine Learning Research*, 17(1):3190–3239, 2016.

[32] Frédéric Dehais, Emrah Akin Sisbot, Rachid Alami, and Mickaël Causse. Physiological and subjective evaluation of a human–robot object hand-over task. *Applied ergonomics*, 42(6):785–791, 2011.

[33] 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.

[34] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

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

[36] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[37] Francisco Gomez-Donoso, Sergio Orts-Escolano, Alberto Garcia-Garcia, Jose Garcia-Rodriguez, John Alejandro Castro-Vargas, Sergiu Ovidiu-Oprea, and Miguel Cazorla. A robotic platform for customized and interactive rehabilitation of persons with disabilities. *Pattern Recognition Letters*, 99:105–113, 2017.

[38] Juan Gómez-Romero, Jesús García, Michael Kandefer, James Llinas, José M Molina, Miguel A Patricio, Michael Prentice, and Stuart C Shapiro. Strategies and techniques for use and exploitation of contextual information in high-level fusion architectures. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[39] Elena Corina Grigore, Kerstin Eder, Anthony G Pipe, Chris Melhuish, and Ute Leonards. Joint action understanding improves robot-to-human object handover. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4622–4629. IEEE, 2013.

[40] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[41] Philipp Gulde and Joachim Hermsdörfer. Smoothness metrics in complex movement tasks. *Frontiers in neurology*, 9, 2018.

[42] Murtaza Hazara and Ville Kyrki. Speeding up incremental learning using data efficient guided exploration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[43] Jacaueline Hemminahaus and Stefan Kopp. Towards adaptive social behavior generation for assistive robots using reinforcement learning. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 332–340. IEEE, 2017.

[44] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. 04 2017.

[45] Heiko Hoffmann, Peter Pastor, Dae-Hyung Park, and Stefan Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, pages 2587–2592. IEEE, 2009.

[46] Chien-Ming Huang, Maya Cakmak, and Bilge Mutlu. Adaptive coordination strategies for human-robot handovers. In *Robotics: science and systems*. Rome, Italy, 2015.

[47] Markus Huber, Claus Lenz, Markus Rickert, Alois Knoll, Thomas Brandt, and Stefan Glasauer. Human preferences in industrial human-robot interactions. In *Proceedings of the international workshop on cognition for technical systems*, 2008.

[48] Markus Huber, Markus Rickert, Alois Knoll, Thomas Brandt, and Stefan Glasauer. Human-robot interaction in handing-over tasks. In *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 107–112. IEEE, 2008.

[49] Markus Huber, Aleksandra Kupferberg, Claus Lenz, Alois Knoll, Thomas Brandt, and Stefan Glasauer. Spatiotemporal movement planning and rapid adaptation for manual interaction. *PloS one*, 8(5):e64982, 2013.

[50] Auke Jan Ijspeert, Jun Nakanishi, Tomohiro Shibata, and Stefan Schaal. Nonlinear dynamical systems for imitation with humanoid robots. In *Proceedings of the IEEE/RAS International Conference on Humanoids Robots (Humanoids2001)*, number CONF, pages 219–226, 2001.

[51] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[52] 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.

[53] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *Advances in neural information processing systems*, pages 575–583, 2013.

[54] Ashesh Jain, Shikhar Sharma, and Ashutosh Saxena. Beyond geometric path planning: Learning context-driven trajectory preferences via sub-optimal feedback. In *Robotics Research*, pages 319–338. Springer, 2016.

[55] Shinya Kajikawa, Takaki Okino, Kohtaro Ohba, and Hikaru Inooka. Motion planning for hand-over between human and robot. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 193–199. IEEE, 1995.

[56] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

[57] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pages 2859–2867, 2013.

[58] Kheng Lee Koay, Emrah Akin Sisbot, Dag Sverre Syrdal, Mick L Walters, Kerstin Dautenhahn, and Rachid Alami. Exploratory study of a robot approaching a person in the context of handing over an object. In *AAAI spring symposium: multidisciplinary collaboration for socially assistive robotics*, pages 18–24, 2007.

[59] Jens Kober and Jan Peters. *Learning motor skills: from algorithms to robot experiments*, volume 97. Springer, 2013.

[60] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[61] Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.

[62] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[63] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 3232–3237. IEEE, 2010.

[64] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345, 2012.

[65] Andras Kupcsik, David Hsu, and Wee Sun Lee. Learning dynamic robot-to-human object handover from human feedback. *arXiv preprint arXiv:1603.06390*, 2016.

[66] 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.

[67] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[68] Przemyslaw A Lasota and Julie A Shah. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human factors*, 57(1):21–33, 2015.

[69] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

[70] Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 156–163. IEEE, 2015.

[71] Jens Lundell, Murtaza Hazara, and Ville Kyrki. Generalizing movement primitives to new situations. In *Annual Conference Towards Autonomous Robotic Systems*, pages 16–31. Springer, 2017.

[72] Eric Martinson, A Huamán Quispe, and Kentaro Oguchi. Towards understanding user preferences in robot-human handovers: How do we decide? In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 516–521. IEEE, 2017.

[73] Martin Mason and Manuel C Lopes. Robot self-initiative and personalization by learning through repeated interactions. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 433–440. ACM, 2011.

[74] José R Medina, Felix Duvallet, Murali Karnam, and Aude Billard. A human-inspired controller for fluid human-robot handovers. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 324–331. IEEE, 2016.

[75] Bernard Michini, Mark Cutler, and Jonathan P How. Scalable reward learning from demonstration. In *2013 IEEE International Conference on Robotics and Automation*, pages 303–308. IEEE, 2013.

[76] Alex Mitrevski, Abhishek Padalkar, Minh Nguyen, and Paul G. Plöger. "Lucy, Take the Noodle Box!": Domestic Object Manipulation Using Movement Primitives and Whole Body Motion. In *Proceedings of the 23rd RoboCup International Symposium*, Sidney, Australia, 2019.

[77] Noriaki Mitsunaga, Christian Smith, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. *Journal of the Robotics Society of Japan*, 24(7):820–829, 2006.

[78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[79] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[80] Krishna Kumar Narayanan, Luis Felipe Posada, Frank Hoffmann, and Torsten Bertram. Scenario and context specific visual robot behavior learning. In *2011 IEEE International Conference on Robotics and Automation*, pages 1180–1185. IEEE, 2011.

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

[82] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. Intention-aware motion planning using learning based human motion prediction. In *Robotics: Science and Systems*, 2017.

[83] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[84] Ugo Pattacini, Francesco Nori, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 1668–1674. IEEE, 2010.

[85] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225. IEEE, 2006.

[86] Jan Peters and Stefan Schaal. Reinforcement learning for parameterized motor primitives. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 73–80. IEEE, 2006.

[87] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[88] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[89] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[90] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE transactions on industrial electronics*, 47(1):140–149, 2000.

[91] Robert Pinsler, Riad Akrour, Takayuki Osa, Jan Peters, and Gerhard Neumann. Sample and feedback efficient hierarchical reinforcement learning from human preferences. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 596–601. IEEE, 2018.

[92] Ana C Huamán Quispe, Eric Martinson, and Kentaro Oguchi. Learning user preferences for robot-human handovers. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 834–839. IEEE, 2017.

[93] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 471–477. IEEE, 2015.

[94] Robin Rasch, Sven Wachsmuth, and Matthias Kcînig. Understanding movements of hand-over between two persons to improve humanoid robot systems. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 856–861. IEEE, 2017.

[95] Silvia Rossi, François Ferland, and Adriana Tapus. User profiling and behavioral adaptation for hri: a survey. *Pattern Recognition Letters*, 99:3–12, 2017.

[96] Leonel Rozo, Sylvain Calinon, Darwin G Caldwell, Pablo Jimenez, and Carme Torras. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, 2016.

[97] Sebastian Schneider and Franz Kummert. Exploring embodiment and dueling bandit learning for preference adaptation in human-robot interaction. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1325–1331. IEEE, 2017.

[98] Satoru Shibata, Kanya Tanaka, and Akira Shimizu. Experimental analysis of handing over. In *Proceedings 4th IEEE International Workshop on Robot and Human Communication*, pages 53–58. IEEE, 1995.

[99] Emrah Akin Sisbot and Rachid Alami. A human-aware manipulation planner. *IEEE Transactions on Robotics*, 28(5):1045–1057, 2012.

[100] Kyle Strabala, Min Kyung Lee, Anca Dragan, Jodi Forlizzi, Siddhartha S Srinivasa, Maya Cakmak, and Vincenzo Micelli. Toward seamless human-robot handovers. *Journal of Human-Robot Interaction*, 2(1):112–132, 2013.

[101] Freek Stulp and Stefan Schaal. Hierarchical reinforcement learning with movement primitives. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 231–238. IEEE, 2011.

[102] Halit Bener Suay and Emrah Akin Sisbot. A position generation algorithm utilizing a biomechanical model for robot-human object handover. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3776–3781. IEEE, 2015.

[103] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[104] Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22, 1992.

[105] Voot Tangkaratt, Herke van Hoof, Simone Parisi, Gerhard Neumann, Jan Peters, and Masashi Sugiyama. Policy search with high-dimensional context variables. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[106] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *2010 IEEE International Conference on Robotics and Automation*, pages 2397–2403. IEEE, 2010.

[107] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *2010 IEEE International Conference on Robotics and Automation*, pages 2397–2403. IEEE, 2010.

[108] Roy M Turner. Context-mediated behavior for intelligent agents. *International Journal of Human-Computer Studies*, 48(3):307–330, 1998.

[109] Roy M Turner, Sonia Rode, and David Gagne. Toward distributed context-mediated behavior for multiagent systems. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 222–234. Springer, 2013.

[110] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[111] Herke Van Hoof, Gerhard Neumann, and Jan Peters. Non-parametric policy search with limited information loss. *The Journal of Machine Learning Research*, 18(1):2472–2517, 2017.

[112] Maxim Vochten, Tinne De Laet, and Joris De Schutter. Generalizing demonstrated motions and adaptive motion generation using an invariant rigid body trajectory representation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 234–241. IEEE, 2016.

[113] Maxim Vochten, Tinne De Laet, and Joris De Schutter. Generalizing demonstrated motions and adaptive motion generation using an invariant rigid body trajectory representation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 234–241. IEEE, 2016.

[114] Michael L Walters, Kerstin Dautenhahn, Sarah N Woods, and Kheng Lee Koay. Robotic etiquette: results from user studies involving a fetch and carry task. In *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 317–324. IEEE, 2007.

[115] Weitian Wang, Rui Li, Zachary Max Diekel, Yi Chen, Zhujun Zhang, and Yunyi Jia. Controlling object hand-over in human–robot collaboration via natural wearable sensing. *IEEE Transactions on Human-Machine Systems*, 49(1):59–71, 2018.

[116] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[117] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[118] Thomas Witzig, J Marius Zöllner, Dejan Pangercic, Sarah Osentoski, Rainer Jäkel, and Rüdiger Dillmann. Context aware shared autonomy for robotic manipulation tasks. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5686–5693. IEEE, 2013.

[119] Akihiko Yamaguchi and Christopher G Atkeson. Neural networks and differential dynamic programming for reinforcement learning problems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5434–5441. IEEE, 2016.

[120] Wayne Zachary, Matthew Johnson, R Hoffman, Travis Thomas, Andrew Rosoff, and Thomas Santarelli. A context-based approach to robot-human interaction. *Procedia Manufacturing*, 3: 1052–1059, 2015.