Data-Efficient Learning of Robotic Grasps From Human Preferences

Dateneffizientes Lernen von Greifbewegungen durch Roboter mittels menschlicher Präferenzen

Master-Thesis von Robert Pinsler aus Berlin

Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Gerhard Neumann

TECHNISCHE UNIVERSITÄT DARMSTADT

Data-Efficient Learning of Robotic Grasps From Human Preferences
Dateneffizientes Lernen von Greifbewegungen durch Roboter mittels menschlicher Präferenzen

Vorgelegte Master-Thesis von Robert Pinsler aus Berlin

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Gerhard Neumann

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 14. August 2017

(Robert Pinsler)

Abstract

The ability to grasp various types of objects from the environment is an important manipulation skill for autonomous robots. It poses a prerequisite for solving many real-world tasks, ranging from object handover to household tasks such as clearing the dish washer. However, the huge variety of objects in our environment presents a major obstacle for robots attempting to acquire grasping skills that match the abilities of humans. Although the field of robotic grasping has received great attention during the last decades, grasping arbitrary objects therefore remains a challenging task.
In contrast to earlier work that was often restricted to analytic approaches in simulation, reinforcement learning techniques allow to learn from grasp experience by trial and error. While the use of reinforcement learning approaches has led to promising results, defining a reward function is still an open problem. Prior work learned a reward model from human feedback for a single grasp type. However, to capture the whole variety of objects in the real world, it is necessary to learn grasping motions across different grasp types.
The goal of this thesis is to devise a reinforcement learning approach that enables the robot to learn how to grasp objects across different grasp types without prior knowledge of the reward function. Because robot operation is both time-consuming and costly, the algorithm should additionally be data-efficient. To achieve this goal, we leverage human feedback to learn multiple grasp policies using a hierarchical reinforcement learning approach. On the upper level, the robot chooses a grasp type and location based on the predicted reward of the resulting trajectory. Subsequently, a grasping motion is generated by the lower-level policy of the selected grasp type. In order to evaluate the outcome of a grasp, we introduce an additional reward model learned entirely from preference feedback. Bayesian optimization and active learning techniques are employed to reduce the number of feedback requests and achieve data-efficient learning.
We examined the usefulness of our approach in various experiments on a ball throwing toy task and a simulated grasping task. We showed that learning an outcome reward model from preferences can improve the performance of the system. Moreover, we were able to reduce the amount of required feedback significantly by introducing an active learning criterion. When applied to robotic grasping, our approach was able to learn how to grasp few known objects using different grasp types. However, it failed when more objects were added to the task. Future work could improve the modeling of the reward function or the generation of grasp locations. Furthermore, additional experiments on a real robotic system are required to evaluate the performance of the proposed approach.

Zusammenfassung

Die Fähigkeit, verschiedenste Arten von Objekten aus der Umgebung greifen zu können, ist für autonome Roboter von zentraler Bedeutung. Sie stellt eine Grundvoraussetzung für das Lösen zahlreicher praktischer Aufgaben wie beispielsweise dem Übergeben von Objekten oder dem Ausräumen eines Geschirrspülers dar. Die enorme Vielfalt an Objekten in unserer Umgebung erschwert es Robotern jedoch deutlich, Greiffähigkeiten zu erlangen, die denen eines Menschen ebenbürtig sind. Obwohl auf dem Gebiet des roboterbasierten Greifens seit Jahrzehnten aktiv geforscht wird, bleibt das Greifen beliebiger Objekte daher eine Herausforderung für Roboter.
Im Gegensatz zu früheren Arbeiten, die oft auf analytische Ansätze in Simulation beschränkt waren, erlauben Methoden des bestärkenden Lernens nach dem Prinzip von Versuch und Irrtum aus Greiferfahrungen zu lernen. Auch wenn die Verwendung dieser Ansätze bereits zu vielversprechenden Ergebnissen geführt hat, bleibt das Definieren einer geeigneten Belohnungsfunktion ein offenes Problem. In einer früheren Arbeit konnte die Belohnungsfunktion für eine einzige Greifart gelernt werden. Um die gesamte Variabilität von Objekten in der Praxis abdecken zu können, ist es jedoch notwendig, Greifbewegungen über verschiedene Greifarten hinweg zu lernen.
Ziel der vorliegenden Arbeit ist es, einen Ansatz auf Basis des bestärkenden Lernens zu entwickeln, der es einem Roboter ermöglicht, Greifbewegungen verschiedener Greifarten zu lernen. Dabei wird angenommen, dass die Bewertungsfunktion vorher nicht bekannt ist. Da der Betrieb eines Roboters sowohl zeit- als auch kostenaufwendig ist, sollte der Ansatz zudem möglichst dateneffizient sein. Um dieses Ziel zu erreichen, nutzen wir menschliches Feedback zum Lernen mehrerer Greifstrategien im Rahmen eines hierarchischen Ansatzes. Auf der oberen Ebene wählt der Roboter zunächst eine Greifart und -position auf Basis der geschätzten Qualität der resultierenden Bewegung aus. Anschließend wird eine konkrete Greifbewegung durch die Greifstrategie der ausgewählten Greifart erzeugt. Um das Resultat eines Greifversuches zu bewerten, lernen wir zusätzlich ein Belohnungsmodell auf Basis von Präferenzfeedback. Darüber hinaus verwenden wir Techniken der Bayes'schen Optimierung und des aktiven Lernens, um die Anzahl von Feedbackanfragen zu reduzieren und dateneffizientes Lernen zu ermöglichen.

Wir haben den Nutzen unseres Ansatzes in zahlreichen Experimenten untersucht. Dabei konnten wir zeigen, dass das Lernen eines zusätzlichen Belohnungsmodells auf Basis von Präferenzen die Performanz des Systems verbessern kann. Darüber hinaus konnten wir die Menge an benötigtem Feedback mithilfe von aktivem Lernen signifikant reduzieren. Angewandt auf eine Greifaufgabe war unser Ansatz in der Lage zu lernen, wie man eine Menge weniger bekannte Objekte mithilfe verschiedener Greifarten greift. Er scheiterte jedoch, wenn die Anzahl der Objekte erhöht wurde. Zukünftige Arbeiten könnten die Modellierung der Belohnungsfunktion oder die Generierung von Kontaktstellen verbessern. Zudem sind weitere Experimente auf einem realen Robotersystem nötig, um die Performanz des vorgestellten Ansatzes weiter zu evaluieren.

Acknowledgments

I am deeply grateful to Prof. Dr. Gerhard Neumann for giving me the opportunity to work on this very interesting topic. His guidance as well as his ongoing support were of great value.
I would also like to thank Prof. Dr. Jan Peters, head of the Intelligent Autonomous Systems group, for providing such a stimulating and productive environment.
Finally, I am deeply indebted to Dr. Riad Akrour and Dr. Takayuki Osa for the many fruitful discussions as well as all the helpful practical advice. I have greatly benefited from their experience.

Contents

Figures

List of Figures

Abbreviations

List of Abbreviations

Notation	Description
ARD	automatic relevance determination
BO	Bayesian optimization
CECER	covariance estimation with controlled entropy reduction
EI	expected improvement
GP	Gaussian process
GP-UCB	Gaussian process upper confidence bound
ICP	iterative closest points
i.i.d.	independently and identically distributed
KL	Kullback-Leibler divergence
MAP	maximum a posteriori
PI	probability of improvement
PS	policy search
REPS	relative entropy policy search
RL	reinforcement learning
UCB	upper confidence bound

List of Algorithms

1 Introduction

1.1 Motivation

Autonomous robots equipped with dexterous manipulation skills yield great potential for improving our lives. For example, household robots could empty the dish washer or hand over everyday items, agricultural robots could pick vegetables, and disaster robots could segregate waste. Such robots need to grasp various types of objects with differences in color, shape, size, mass, and so on. Some of those objects might not even have been seen before by the robot. Due to that, robots have to be able to perform a broad range of grasping motions varying from object to object. It therefore does not suffice to only memorize or pre-program a limited number of grasps. Likewise, it is not possible to preconceive all ways of executing grasps in advance. Robots instead need to be able to generalize their knowledge to new situations. But although grasping has been a long-standing research area in the robotics community, grasping arbitrary objects remains a challenging problem.

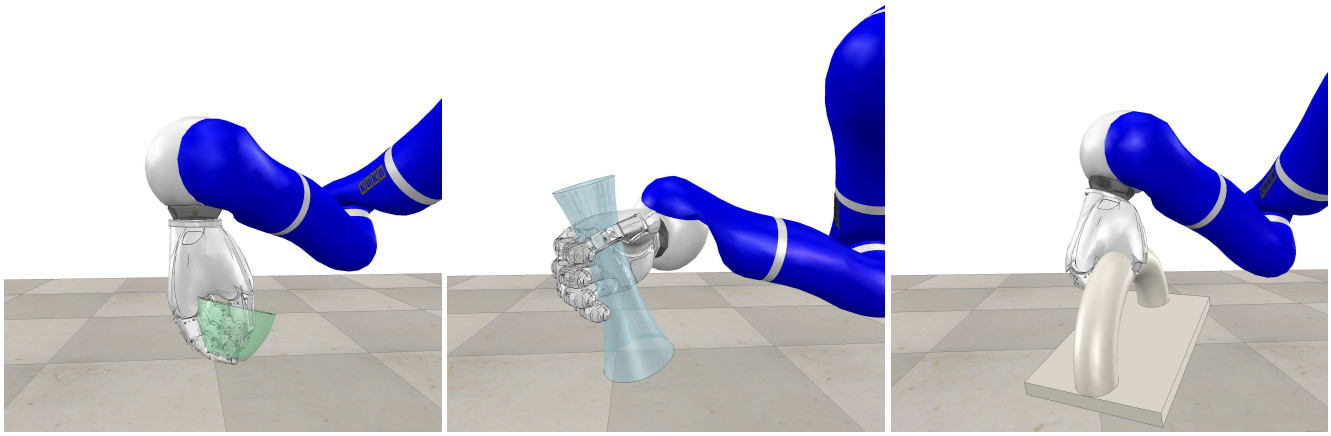
Robotic grasping approaches [8, 9] can be divided into two different categories: analytic and data-driven approaches. Analytic approaches attempt to model the interaction between the object and the robot end-effector directly through the laws of physics [10]. Much of the research in this area is concerned with constructing force-closure grasps for multi-fingered hands, where a grasp is said to be force-closure if the fingers gripping the object are able to resist any external forces that act on the object [11]. The task of finding contact points that lead to a force-closure grasp is also referred to as grasp synthesis, whereas the evaluation of grasps is called grasp analysis. Shimoga [11] identifies four important properties of force-closure grasps, namely

- dexterity: the finger joints are configured in such a way that they can achieve a secondary objective,
- equilibrium: the fingertip forces are chosen such that the external forces and moments acting on the object are counter-balanced,
- stability: the grasp can resist external disturbances, and
- dynamic behavior: the grasp configuration allows for a dynamic behavior of the object that is in line with the task of the robot.

Grasp synthesis is therefore often framed as a constrained optimization problem, where the goal is to find a force-closure grasp that maximizes a set of quality measures subject to geometric, kinematics and dynamics constraints. Various quality measures have been proposed that quantify these properties [12, 13]. Many of them analyze the contact forces applied to the object. For instance, the popular ϵ-metric [14] considers the radius of the largest sphere contained in the convex hull of the grasp wrench space, where the grasp wrench space is the space of feasible wrenches at the contact points of the fingers. Such an analysis is particularly suited for precision grasps, which are grasps where the object is gripped with the fingertips only. In fact, most of the past research in this area focused on precision grasps [11].

However, empirical studies of humans have shown that multiple grasp types are required to grasp arbitrary objects [10, 15]. In industrial settings, different simpler end-effectors are used depending on the task. This mode of operation works well if the same task is repeatedly executed over a longer period of time. If the environment is less constrained, more flexibility is usually required. In fact, humans even choose between different grasp types for the same object depending on the task they try to achieve. For example, to take a bottle out of a bottle case one would grip the bottle at the neck of the bottle with the fingertips (precision grasp), whereas one would completely enclose the bottle with the fingers to transport it (power grasp). Although we do not focus on task requirements in this thesis, being able to perform different grasp types is clearly an important skill. See Fig. 1.1 for some examples of different grasp types.

Analytic approaches require precise physical and geometric models of the environment, which are only available in simulation. Furthermore, they make several simplifying assumptions to keep the computations tractable, e.g. the use of rigid-body models, linearized kinematics and quasi-static environments [10]. Data-driven approaches try to avoid these shortcomings by making use of some form of grasp experience to synthesize grasps. Grasps are usually selected from a grasp database or grasp candidates are sampled and ranked based on a quality measure. In order to be able to grasp previously unseen objects, objects are commonly represented and compared in terms of perceptual features. While most of these approaches assume that a complete grasp database is given or can be pre-computed offline, reinforcement learning (RL) allows to collect grasp experiences from trial and error. The grasp dataset is thus autonomously built and adapted in an online fashion. [8]



(a) Pinch grasp **(b)** Power grasp **(c)** Medium wrap grasp

Figure 1.1: Examples of different grasp types.

Krömer et al. [16] decompose a grasping task into two sub-problems: (1) determine where to grasp an object, and (2) synthesize and finally execute a suitable grasp. The first problem requires to process the sensory input about the scene in order to select a grasp location. Given a grasp location, the problem of generating a grasping motion can then be framed as a reinforcement learning task, where the goal is to learn a grasping policy from trial and error. Such a strategy is similar to how infants learn to grasp [17]. But in contrast to humans, robots cannot collect years of experience to acquire grasping skills. Robot operation during learning is both time-consuming and causes wear and tear. It is therefore important that the robot learns to grasp in a data-efficient manner.

Learning a grasping policy through RL requires the definition of a reward function. As discussed earlier, most of the proposed quality measures were motivated by grasp analysis in simulation. But the outcome of a grasp in a real dynamic and noisy environment can be very different from what was planned. In fact, Balasubramanian et al. [18] showed that grasps produced by optimizing such traditional grasp quality measures performed worse than kinesthetic teach-in on real systems. Moreover, Roa and Suárez [13] point out that there is no single measure suitable for every situation. That is because those metrics only encode certain criteria of a good grasp, e.g. stability or dexterity. While more complex reward functions could be conceived, such reward shaping practices can quickly lead to undesirable behaviors of the robot [19]. These findings suggest that designing a reward function by hand is not trivial.

Alternatively, inverse reinforcement learning [20, 21] can be used to recover the underlying reward function from expert demonstrations. Demonstrations are usually provided by humans through tele-operation or kinesthetic teach-in. Presenting good or even optimal grasps can however be challenging without prior training. In particular, differences between the human hand and robot hardware complicate this task. As we do not want to make the assumption that the human already is an expert in demonstrating grasps, we refrain from using inverse reinforcement learning techniques.

Learning from human feedback is another viable alternative. Instead of having to demonstrate whole trajectories, humans merely need to assess the quality of a grasp. An intuitive evaluation scheme is to rate each rollout on a categorical or continuous scale. However, ratings on a scale are problematic as human judgements are subject to biases such as anchoring or drift [22]: anchoring is the tendency to overweight information presented at the beginning; drift refers to changing one's evaluation scheme over time, e.g. because the weighting of different aspects is adapted. Furthermore, the scale can vary widely among observers. On the contrary, humans are particularly apt at comparing items [23, 24], making preference feedback more reliable than absolute feedback. We take advantage of this fact by learning a reward model from preference feedback about different grasps. To still allow for the possibility of noisy feedback, a probabilistic reward model is employed.

The goal of this thesis is to design a data-efficient reinforcement learning approach that allows the robot to learn how to grasp objects without a known reward function at hand. The grasping motions should be versatile enough to cover a wide range of objects, including previously unseen ones. We therefore propose a hierarchical reinforcement learning framework that is able to learn grasps across different grasp types. By introducing different grasp types, the versatility of grasping motions is broken down into multiple action policies to be learned. Grasp generation is thus performed in two steps. First, an upper-level policy chooses a grasp type and grasp location based on the expected grasp quality for the given object. The grasp location is determined from point cloud information of the object. Second, a grasping motion is generated from the lower-level policy of the selected grasp type.

According to the proposed architecture, grasp quality needs to be evaluated in two situations. First, the expected reward of using a certain grasp type at a grasp location needs to be estimated before each rollout. Second, the quality of the actual grasp trajectory has to be evaluated. These two estimates can vary significantly due to system noise and uncertainty about

the inputs. We therefore learn two types of reward models: (1) an outcome reward model that assesses the outcome of rollouts based on human preferences, and (2) a context-parameter reward model for each grasp type to predict the quality of a grasp location and motion parameters generated by a grasping policy. The context-parameter reward models are learned from the estimated rewards of the outcome reward model. Hence, we avoid to learn the high-dimensional context-parameter reward models directly from preferences. In contrast to that, the outcome reward model can usually be learned much faster since outcomes are low-dimensional representations of the executed trajectory.

Finally, data-efficiency can be achieved through Bayesian optimization (BO) and active learning techniques. In particular, we learn probabilistic reward models that allow to consider uncertainty about the rewards when a grasp is selected. Moreover, we reduce the number of feedback requests by using active learning to decide when to query the human.

1.2 Related Work

Our work is related to several approaches from different fields, including learning to grasp from trial and error, reward learning and active learning. In the following, relevant prior work in these areas is briefly reviewed and compared to the proposed approach of this thesis.

Learning from trial and error Instead of relying on a given dataset, several authors acquire or refine a grasp experience dataset through trial and error. See Fig. 1.2 for some examples. Herzog et al. [25] learn feasible grasp configurations from human demonstrations and rank them for grasping new objects by matching local object shape templates. They use grasping feedback to improve the matching. However, new grasp hypotheses can only be learned from additional human demonstrations. Stulp et al. [26] use reinforcement learning to learn the goal and shape parameters of a movement primitive. They achieve robust grasps by taking position uncertainty into consideration. Krömer et al. [16] propose a hierarchical architecture comprising of an upper-level reinforcement learner for predicting the grasp location and a low-level reactive controller that generates the grasp motion.

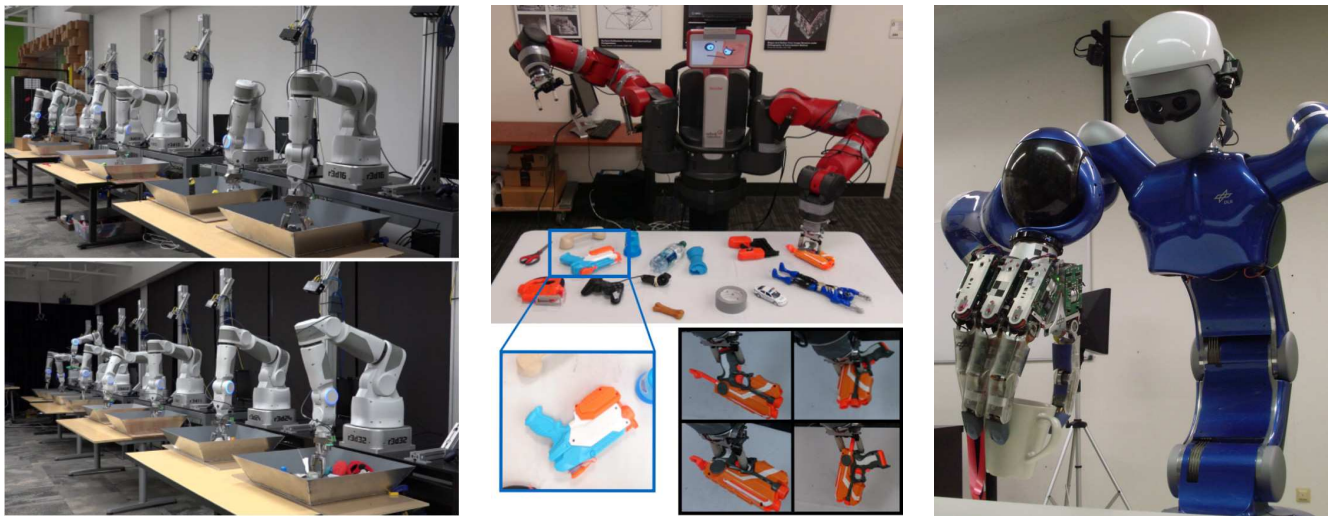
In order to reduce complexity, these methods often use a parallel jaw gripper as the end-effector of the robot. This design choice limits the graspable objects to those with sufficiently parallel and plane shapes [11]. Therefore, the robot is often restricted to perform only certain grasp types, e.g. vertical pinch grasps. However, different grasp strategies are required depending on the object to grasp [10]. Osa et al. [7] propose a hierarchical reinforcement learning architecture that is able to generalize grasps to multiple grasp types and objects. They extract several candidate grasp locations for different grasp types by matching them with local contact parts of previously successful grasps. Given the grasp locations, they use an upper-level policy to select a grasp type and location based on predicted rewards. Subsequently, the grasp is executed by a grasp type specific lower-level policy. We use the same basic architecture as Osa et al. [7], but deviate in the way the rewards are learned.

With the advent of deep learning techniques in the context of grasping, self-supervised learning techniques have been proposed for large-scale data acquisition more recently [1, 2]. For instance, Levine et al. [1] trained a CNN grasp prediction model from over 800,000 grasp trials collected by up to 14 robots working in parallel. While these works are a promising step towards scaling robotic grasping applications, this thesis takes a different direction as we strive for data-efficiency.

Reward learning Osa et al. [7] use GP regression [4] to predict the quality of a grasp and employ the popular ϵ-metric [14] to evaluate the actual grasp quality. However, this metric only considers grasp stability, which is necessary but not sufficient for a good grasp. Furthermore, analytic measures such as the ϵ-metric require to determine the exact contact locations, which is only possible in simulation. Daniel et al. [27] instead learn a reward function from outcomes based on absolute human feedback. Our work can be seen as a combination of these two approaches, where both the expected grasp quality and the actual reward from an outcome are learned. However, in contrast to [27], we ask for preference feedback in favor of the less reliable absolute ratings.

Preference-based reward learning has also been used for various other reinforcement learning tasks [28, 29, 30, 31, 32].

Active learning In order to reduce the number of required rollouts, active learning can be used to decide which grasp candidate to try next. Morales et al. [33] predict grasp success probabilities from visual features using a k-nearest neighbor approach. They select grasp candidates based on where the classifier has low predictive confidence, encouraging the system to explore regions with high uncertainty. Montesanoa and Lopes [34] follow a Bayesian approach and use beta–binomial distributions to model the predictive distribution over grasp success probability. Given the mean and variance of this distribution, they select the next grasp candidate. Similarly, the authors in [16, 7] employ a GP regression model to estimate the predictive reward distribution, which they then integrate into an active learning approach. Daniel et al. [27] use an active learning criterion to decide whether to request human feedback about a grasp. In particular, they propose to request feedback when the expected information gain with respect to the grasping policy is large. We employ a similar strategy, but only consider the information gain with respect to the reward model.



(a) (b) (c)

Figure 1.2: Robotic grasping examples. **(a)**: Large-scale data collection across 14 robotic manipulators picking up objects with a parallel jaw gripper [1]. **(b)**: Baxter robot grasps an item with irregular shape in a cluttered scene. As shown in the bottom right box, the robot is able to grasp the object in different ways [2]. **(c)**: Justin robot with five-fingered hand successfully lifts a mug [3].

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces basic concepts and preliminary notation to the reader. It covers Gaussian processes (Section 2.1), Bayesian optimization and active learning (Section 2.2), and policy search for robotics (Section 2.3). In Chapter 3, the hierarchical reinforcement learning approach is described and applied to robotic grasping. After a general overview, Section 3.1 elaborates on how to learn the reward models, Section 3.2 explains how to learn the policies, and Section 3.3 tailors the approach towards robotic grasping. Chapter 4 presents the experimental setup for two tasks and evaluates the results. In particular, we perform experiments for a ball throwing toy task in Section 4.1, followed by a simulated grasping task in Section 4.2. The chapter closes with a discussion of the results in Section 4.3. Finally, Chapter 5 concludes this thesis and outlines future work.

2 Background

This chapter serves as an introduction to several important topics that are covered in this thesis. Section 2.1 describes Gaussian processes in more detail. The concepts of Bayesian optimization and active learning are introduced in Section 2.2. Lastly, the policy search framework in the context of robotics is presented in Section 2.3.

2.1 Gaussian Processes

GPs are non-parametric statistical models that allow to maintain a distribution over functions. In this thesis, we are particularly interested in using GPs for regression and preference learning. The two GP models are subsequently introduced in Section 2.1.1 and 2.1.2.

2.1.1 Gaussian Processes for Regression

Regression is a supervised learning problem, where the goal is to infer the functional relationship between inputs $\mathbf{x} \in \mathscr{R}^d$ and real-valued outputs $y \in \mathscr{R}$ given a dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i, y_i\}_{i=1,\dots,N}$, for example to make predictions about the output value at a certain input location. We assume the observed output y is affected by independently and identically distributed (i.i.d.) Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ with variance σ_n^2, i.e.

$$y = f(\mathbf{x}) + \epsilon, \tag{2.1}$$

where $f : \mathscr{R}^d \mapsto \mathscr{R}$ is a real-valued function. Having introduced the basic problem, the GP regression model is formally described in the following. The majority of this section is based on [4].

Gaussian Process Regression Model

A GP is a collection of random variables, of which any finite subset is jointly Gaussian distributed [4]. It is sufficiently described by its mean function $m(\mathbf{x}) = \mathbb{E}[f(x)]$ and covariance function $k(\mathbf{x}, \mathbf{x}') = \mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}'))$,

$$f(\mathbf{x}) \sim \mathrm{GP}^k(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

where the mean function is usually assumed to be zero, i.e. $m(\mathbf{x}) = \mathbf{0}$. The most widely used covariance function is the squared exponential covariance function with automatic relevance determination (ARD),

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left((\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')\right), \tag{2.2}$$

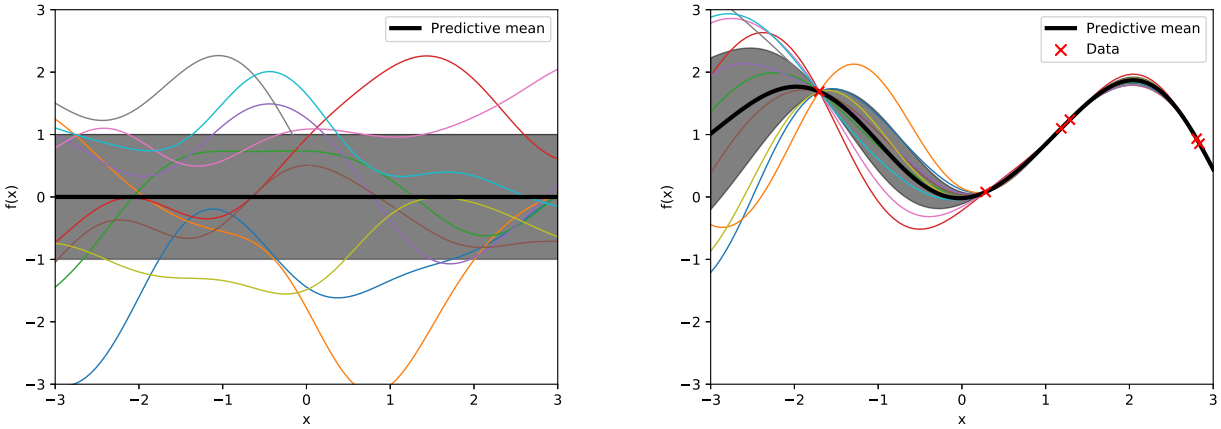
where $\mathbf{M} = \mathrm{diag}(\boldsymbol{\ell})$ is a diagonal matrix containing positive bandwidth parameters $\boldsymbol{\ell}$, which determine the contribution of each dimension of the input. Together with the function variance σ_f^2, they form the hyperparameters of the kernel.
The GP treats the function values as random variables and therefore implicitly defines a prior distribution $p(\mathbf{f}|\mathbf{X})$ over function values $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^T$,

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K}) = |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp\left(\mathbf{f}^T \mathbf{K}^{-1} \mathbf{f}\right),$$

where the ij-th element of the covariance matrix $\mathbf{K}$ is defined by $k(\mathbf{x}_i, \mathbf{x}_j)$. Given some likelihood $p(\mathbf{y}|\mathbf{f})$, the posterior over latent function values can be obtained through Bayes' rule, i.e.

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})},$$

where $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f}$ is a normalization term. For example, the regression model in Eq. 2.1 is expressed by the likelihood $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbf{I})$, where $\mathbf{I}$ is the identity matrix. Because of the Gaussian form of the likelihood, the integral over the latent function values in the normalization term can be solved analytically.
Next, it is explained how to make predictions with this model. Note that in the following we omit the dependence on the input data $\mathbf{X}$ to improve readability.



(a) Prior **(b)** Posterior

Figure 2.1: Example of Gaussian process regression with mean prediction and uncertainty estimate, where error bars denote single standard deviation. Colored lines represent function samples. **(a)**: The GP starts with a zero mean Gaussian prior and large uncertainty about the predictions. **(b)**: Once data is observed, the posterior is updated and the uncertainty is reduced at the input locations. The figure is based on [4].

Prediction

To make predictions at some test location $\mathbf{x}^*$, we first realize that the prior function values $\mathbf{f}$ and the prior function value f^* of the test input $\mathbf{x}^*$ are jointly Gaussian distributed, i.e.

$$\begin{bmatrix} \mathbf{f} \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \boldsymbol{\kappa} \\ \boldsymbol{\kappa}^T & \kappa \end{bmatrix}\right), \tag{2.3}$$

where $\boldsymbol{\kappa} = [k(\mathbf{x}_1, \mathbf{x}^*), \ldots, K(\mathbf{x}_N, \mathbf{x}^*)]^T$ and $\kappa = k(\mathbf{x}^*, \mathbf{x}^*)$. Similarly, the conditional distribution $p(f^*|\mathbf{f})$ is Gaussian. The predictive distribution is thus given by

$$p(f^*|\mathbf{y}) = \int p(f^*|\mathbf{f}) p(\mathbf{f}|\mathbf{y}) d\mathbf{f},$$

which is also a Gaussian distribution $\mathcal{N}(\mu_*, \sigma_*^2)$ with

$$\mu_* = \boldsymbol{\kappa}^T (\mathbf{K}_y)^{-1} \mathbf{f}, \tag{2.4a}$$
$$\sigma_*^2 = \kappa - \boldsymbol{\kappa}^T (\mathbf{K}_y)^{-1} \boldsymbol{\kappa}, \tag{2.4b}$$

where $\mathbf{K}_y = \mathbf{K} + \sigma_n^2 \mathbf{I}$ is the covariance matrix for the output values $\mathbf{y}$. Fig. 2.1 shows an example of a GP regression model, where the predictive distribution is altered as more data is observed.

Model Selection

Until now, we have neglected the dependence on the hyperparameters $\boldsymbol{\theta} = \{\sigma_f, \sigma_n, \ell\}$. Following a full Bayesian treatment requires to integrate out all hyperparameters $\boldsymbol{\theta}$ for prediction, i.e.

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}^*|\mathbf{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta},$$

where we explicitly include the dependence on $\boldsymbol{\theta}$. The parameter posterior $p(\boldsymbol{\theta}|\mathbf{y})$ is given by

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y})} \tag{2.5}$$

where $p(\mathbf{y}) = \int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$, and $p(\boldsymbol{\theta})$ denotes a prior distribution of the hyperparameters, which is also referred to as a hyperprior. The integral in Eq. 2.5 is only analytically tractable for certain choices of $p(\boldsymbol{\theta})$. In general, Markov chain Monte Carlo or approximate methods have to be used. We resort to determining an optimal point estimate for $\boldsymbol{\theta}$ by maximizing the marginal likelihood, i.e. $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{y}|\boldsymbol{\theta})$. In practice, the log marginal likelihood is maximized instead, which for the regression case is given by

$$\log p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{d}{2}\log 2\pi, \tag{2.6}$$

where d is the dimensionality of the data. The first term is the only one involving the target values $\mathbf{y}$ and can therefore be interpreted as the data-fit term, whereas the second term serves as a complexity penalty with respect to the covariance function [4]. The partial derivatives of Eq. 2.6 with respect to the hyperparameters $\boldsymbol{\theta}$ are as follows:

$$\frac{\partial \log p(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathbf{K}_y^{-1}\right)\frac{\partial \mathbf{K}_y}{\partial \theta_j}\Big),$$

where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{y}$. Given the derivatives, gradient-based optimization techniques can then be used to estimate the hyperparameters.

2.1.2 Gaussian Processes for Preference Learning

It is possible to use GPs for learning from human preferences over inputs. In such a preference learning setup [35], we are given N distinct instances $\mathscr{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and M preference relations $\mathscr{D} = \{\mathbf{v}_1 \succ \mathbf{u}_1, \ldots, \mathbf{v}_M \succ \mathbf{u}_M\}, \mathbf{v}_k \in \mathscr{X}, \mathbf{u}_k \in \mathscr{X}$ over those instances, where $\mathbf{v}_k \succ \mathbf{u}_k$ denotes that $\mathbf{v}_k$ is preferred to $\mathbf{u}_k$.
Chu and Ghahramani [36] proposed a likelihood function specifically designed to learn a probabilistic model from such preferences. In the following, this model is presented in more detail.

Gaussian Process Preference Learning Model

A common approach towards preference learning is to assume that there exists an underlying utility function $f(\mathbf{x})$ that determines the ordering of the instances $\mathbf{x}_i$ [35]. In the GP framework, we place a zero-mean prior $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$ over these latent function values, where $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$, and the ij-th element of the covariance matrix $\mathbf{K}$ is determined by the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. In particular, the squared exponential covariance function with ARD is used as defined in Eq. 2.2.
Ideally, the latent function values $\mathbf{f}$ would be consistent with the preference relations in $\mathscr{D}$, such that $f(\mathbf{v}_k) > f(\mathbf{u}_k)$ implies $\mathbf{v}_k \succ \mathbf{u}_k$. However, in practice human preferences can be noisy. To account for such circumstances, it is assumed that the preference feedback follows a noisy utility function $y(\cdot)$, i.e.

$$y(\mathbf{v}_k) = f(\mathbf{v}_k) + \epsilon_v \qquad y(\mathbf{u}_k) = f(\mathbf{u}_k) + \epsilon_u,$$

with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. This probabilistic formulation is also known as the Thurstone-Mosteller model [37, 24]. The likelihood function for a preference relation under this model is given by

$$p\left(\mathbf{v}_k \succ \mathbf{u}_k | f(\mathbf{v}_k), f(\mathbf{u}_k)\right) = p\left(y(\mathbf{v}_k) > y(\mathbf{u}_k) | f(\mathbf{v}_k), f(\mathbf{u}_k)\right) = \Phi\left(z_k\right), \tag{2.7}$$

where

$$z_k = \frac{f(\mathbf{v}_k) - f(\mathbf{u}_k)}{\sqrt{2}\sigma_n},$$

and $\Phi(\cdot)$ is the standard normal cumulative density function. Assuming i.i.d. data, the likelihood $p(\mathscr{D}|\mathbf{f})$ factorizes as follows,

$$p(\mathscr{D}|\mathbf{f}) = \prod_{k=1}^{M} p\left(\mathbf{v}_k \succ \mathbf{u}_k | f(\mathbf{v}_k), f(\mathbf{u}_k)\right), \tag{2.8}$$

and the posterior over latent function values is given by

$$p(\mathbf{f}|\mathscr{D}) = \frac{p(\mathscr{D}|\mathbf{f})p(\mathbf{f})}{p(\mathscr{D})}.$$

However, in contrast to the regression case in Section 2.1.1, the integral in the marginal likelihood $p(\mathscr{D}) = \int p(\mathscr{D}|\mathbf{f})p(\mathbf{f})d\mathbf{f}$ is not analytically tractable due to the non-Gaussian likelihood. Therefore, it is necessary to use approximate or Markov chain Monte Carlo techniques. In particular, we perform a Laplace approximation at the maximum a posteriori (MAP) estimate $\mathbf{f}_{\text{MAP}} = \arg\max_f p(\mathbf{f}|\mathscr{D})$, which approximates the posterior as a Gaussian centered on the MAP estimate [38]. Maximizing the posterior with respect to $\mathbf{f}$ yields the same MAP estimate $\mathbf{f}_{\text{MAP}}$ as minimizing the un-normalized log posterior, i.e. $\arg\max_f p(\mathbf{f}|\mathscr{D}) = \arg\min_f -\log p(\mathscr{D}|\mathbf{f}) - \log p(\mathbf{f})$. Using the GP prior $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, we define

$$S(\mathbf{f}) = -\log p(\mathscr{D}|\mathbf{f}) + \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} + c,$$

where c comprises any terms independent of $\mathbf{f}$. Given the likelihood $p(\mathscr{D}|\mathbf{f})$ from Eq. 2.8, minimizing the functional $S(\mathbf{f})$ with respect to $\mathbf{f}$ is a convex optimization problem [36]. The first and second order derivatives are given by

$$\nabla S(\mathbf{f}) = \nabla - \log p(\mathscr{D}|\mathbf{f}) + \mathbf{K}^{-1}\mathbf{f}, \tag{2.9a}$$

$$\nabla^2 S(\mathbf{f}) = \nabla^2 - \log p(\mathscr{D}|\mathbf{f}) + \mathbf{K}^{-1}, \tag{2.9b}$$

where ∇ is the gradient operator. The derivatives of the negative log likelihood $-\log p(\mathscr{D}|\mathbf{f}) = -\sum_{k=1}^{M} \log \Phi(z_k)$ are as follows:

$$\boldsymbol{\beta} = \frac{\partial - \sum_{k=1}^{M} \log \Phi(z_k)}{\partial f(\mathbf{x}_i)} = \sum_{k=1}^{M} \frac{-s_k(\mathbf{x}_i)}{\sqrt{2}\sigma_n} \frac{\phi(z_k)}{\Phi(z_k)},$$

$$\boldsymbol{\Gamma} = \frac{\partial^2 - \sum_{k=1}^{M} \log \Phi(z_k)}{\partial f(\mathbf{x}_i)\partial f(\mathbf{x}_j)} = \sum_{k=1}^{M} \frac{s_k(\mathbf{x}_i)s_k(\mathbf{x}_j)}{2\sigma_n^2} \left(\frac{\phi^2(z_k)}{\Phi^2(z_k)} + \frac{z_k\phi(z_k)}{\Phi(z_k)} \right), \qquad s_k(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{x} = \mathbf{v}_k \\ -1 & \text{if } \mathbf{x} = \mathbf{u}_k \\ 0 & \text{else} \end{cases}$$

where $\phi(\cdot)$ is the standard normal density function. From $\nabla S(\mathbf{f})|_{\mathbf{f}=\mathbf{f}_{\text{MAP}}} = 0$ in combination with Eq. 2.9a it follows that the solution is given by

$$\mathbf{f}_{\text{MAP}} = \mathbf{K}\boldsymbol{\beta}_{MAP}.$$

As the solution is not in closed form, Newton's method can be used to iteratively minimize $S(\mathbf{f})$,

$$\mathbf{f}' = \mathbf{f} - \left(\nabla^2 S(\mathbf{f})\right)^{-1} \nabla S(\mathbf{f}).$$

Given the MAP estimate, a second-order Taylor expansion of $S(\mathbf{f})$ around $\mathbf{f}_{\text{MAP}}$ is performed, i.e.

$$S(\mathbf{f}) \approx S(\mathbf{f}_{\text{MAP}}) - \frac{1}{2}(\mathbf{f} - \mathbf{f}_{\text{MAP}})^T \mathbf{H}_{\text{MAP}} (\mathbf{f} - \mathbf{f}_{\text{MAP}}),$$

where $\mathbf{H}_{\text{MAP}} = \nabla^2 S(\mathbf{f})|_{\mathbf{f}=\mathbf{f}_{\text{MAP}}} = \mathbf{K}^{-1} + \boldsymbol{\Gamma}_{\text{MAP}}$ is the Hessian of $S(\mathbf{f})$ at $\mathbf{f}_{\text{MAP}}$. The posterior $p(\mathbf{f}|\mathscr{D})$ can then be approximated as a Gaussian,

$$p(\mathbf{f}|\mathscr{D}) = \frac{1}{Z_p} p(\mathbf{f}, \mathscr{D}) \approx \frac{1}{Z_q} q(\mathbf{f}, \mathscr{D}) \tag{2.10a}$$

$$= \frac{1}{Z_q} p(\mathbf{f}_{\text{MAP}}, \mathscr{D}) \exp\left(-\frac{1}{2}(\mathbf{f} - \mathbf{f}_{\text{MAP}})^T \mathbf{H}_{\text{MAP}} (\mathbf{f} - \mathbf{f}_{\text{MAP}}) \right) \tag{2.10b}$$

$$= \mathcal{N}(\mathbf{f}_{\text{MAP}}, (\mathbf{K}^{-1} + \boldsymbol{\Gamma}_{\text{MAP}})^{-1}), \tag{2.10c}$$

where $p(\mathbf{f}_{\text{MAP}}, \mathscr{D}) = \exp(-S(\mathbf{f}_{\text{MAP}}))$, $Z_p = p(\mathscr{D})$, and Z_q can be found through inspection.

Prediction

Given the model, it is possible to predict the preference relation between two instances $\mathbf{r} \in \mathscr{X}, \mathbf{s} \in \mathscr{X}$ using the predictive preference $p(\mathbf{r} \succ \mathbf{s}|\mathscr{D})$ that itself depends on the predictive distribution $p(\mathbf{f}^*|\mathscr{D})$. Therefore, we first derive the expression for the predictive distribution and then provide a solution for the predictive preference.
Similar to Eq. 2.3, the prior latent function values $\mathbf{f}$ and $\mathbf{f}^* = [f(\mathbf{q}), f(\mathbf{r})]^T$ are jointly Gaussian distributed,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \boldsymbol{\kappa} \\ \boldsymbol{\kappa}^T & \mathbf{K}_* \end{bmatrix} \right), \tag{2.11}$$

where $\boldsymbol{\kappa} = \begin{bmatrix} k(\mathbf{r},\mathbf{x}_1),\ldots,k(\mathbf{r},\mathbf{x}_N) \\ k(\mathbf{s},\mathbf{x}_1),\ldots,k(\mathbf{s},\mathbf{x}_N) \end{bmatrix}^T$ and $\mathbf{K}_* = \begin{bmatrix} k(\mathbf{r},\mathbf{r}) & k(\mathbf{r},\mathbf{s}) \\ k(\mathbf{s},\mathbf{r}) & k(\mathbf{s},\mathbf{s}) \end{bmatrix}$.
Thanks to the Laplace approximation, the approximate posterior $p(\mathbf{f}|\mathscr{D})$ is Gaussian (see Eq. 2.10a), and the predictive distribution $p(\mathbf{f}^*|\mathscr{D})$ can be computed analytically,

$$p(\mathbf{f}^*|\mathscr{D}) = \int p(\mathbf{f}^*|\mathbf{f})p(\mathbf{f}|\mathscr{D})d\mathbf{f} = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*),$$

yielding a Gaussian distribution with mean $\boldsymbol{\mu}^*$ and covariance $\boldsymbol{\Sigma}^*$,

$$\boldsymbol{\mu}^* = [\mu_r^*, \mu_s^*] = \boldsymbol{\kappa}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}},$$
$$\boldsymbol{\Sigma}^* = \begin{bmatrix} \Sigma_{rr}^* & \Sigma_{rs}^* \\ \Sigma_{sr}^* & \Sigma_{ss}^* \end{bmatrix} = \mathbf{K}_* - \boldsymbol{\kappa}^T(\mathbf{K} + \boldsymbol{\Gamma}_{\text{MAP}}^{-1})^{-1}\boldsymbol{\kappa}$$

analogous to Eq. 2.4. Finally, the predictive preference $p(\mathbf{r} \succ \mathbf{s}|\mathscr{D})$ is given by

$$p(\mathbf{r} \succ \mathbf{s}|\mathscr{D}) = \int p(\mathbf{r} \succ \mathbf{s}|\mathbf{f}^*, \mathscr{D})p(\mathbf{f}^*|\mathscr{D})d\mathbf{f}^* = \Phi\left(\frac{\mu_r^* - \mu_s^*}{\sigma_*}\right),$$

where $\sigma_*^2 = 2\sigma_n^2 + \Sigma_{rr}^* + \Sigma_{ss}^* - \Sigma_{rs}^* - \Sigma_{sr}^*$.

Model Selection

Similar to the regression case, the hyperparameters $\boldsymbol{\theta} = \{\sigma_f, \sigma_n, \boldsymbol{\ell}\}$ are estimated by maximizing the marginal likelihood $p(\mathscr{D}|\boldsymbol{\theta}) = \int p(\mathscr{D}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f}$. Again, the integral over the latent function values $\mathbf{f}$ is not analytically tractable because the likelihood (Eq. 2.8) is not conjugate to the GP prior. However, we can re-use results from the Laplace approximation (Eq. 2.10a) to obtain an analytic expression:

$$p(\mathscr{D}|\boldsymbol{\theta}) = \int p(\mathscr{D},\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \approx \int q(\mathscr{D},\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \tag{2.12a}$$
$$= \exp(-S(\mathbf{f}_{\text{MAP}})) \int \exp\left(-\frac{1}{2}(\mathbf{f}-\mathbf{f}_{\text{MAP}})^T\mathbf{H}_{\text{MAP}}(\mathbf{f}-\mathbf{f}_{\text{MAP}})\right)d\mathbf{f} \tag{2.12b}$$
$$= |\mathbf{I} + \mathbf{K}\boldsymbol{\Gamma}_{\text{MAP}}|^{-\frac{1}{2}} \exp(-S(\mathbf{f}_{\text{MAP}})), \tag{2.12c}$$

where $\mathbf{I}$ is the identity matrix. Eq. 2.12c can then be maximized with respect to $\boldsymbol{\theta}$ using gradient-based optimization. In practice, it is advantageous to optimize the log hyperparameters to ensure positivity. The gradients of the log marginal likelihood with respect to the log hyperparameters are given by [36]

$$\frac{\partial \log p(\mathscr{D}|\boldsymbol{\theta})}{\partial \log \lambda_j} = \frac{\lambda_j}{2}\mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \lambda_j}\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \frac{\lambda_j}{2}\text{Tr}\left(\mathbf{H}_{\text{MAP}}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \lambda_j}\boldsymbol{\Gamma}_{\text{MAP}}\right) - \frac{\lambda_j}{2}\text{Tr}\left(\mathbf{H}_{\text{MAP}}\frac{\partial \boldsymbol{\Gamma}_{\text{MAP}}}{\partial \lambda_j}\right),$$
$$\frac{\partial \log p(\mathscr{D}|\boldsymbol{\theta})}{\partial \log \sigma_n} = \sigma_n\left(\frac{\partial \sum_{k=1}^M \log\Phi\left(\frac{f_{\text{MAP}}(\mathbf{v}_k) - f_{\text{MAP}}(\mathbf{u}_k)}{\sqrt{2}\sigma_n}\right)}{\partial \sigma_n}\right) - \frac{\sigma_n}{2}\text{Tr}\left(\mathbf{H}_{\text{MAP}}\frac{\partial \boldsymbol{\Gamma}_{\text{MAP}}}{\partial \sigma_n}\right),$$

where $\boldsymbol{\lambda} = \{\sigma_f, \boldsymbol{\ell}\}$ are the hyperparameters of the kernel.

2.2 Bayesian Optimization and Active Learning

Bayesian optimization and active learning are two closely related fields that offer useful tools for data-efficient learning. Both approaches are reviewed in this section. In particular, Section 2.2.1 introduces the BO framework while active learning is covered subsequently in Section 2.2.2.

2.2.1 Bayesian Optimization

BO [5] is concerned with globally optimizing a possibly non-convex, real-valued function $\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$. It is assumed that the objective is a black-box function that only returns a noisy sample when evaluated at some query point $\mathbf{x}$. Typically, BO is used in settings where function evaluations are expensive and therefore efficient sampling is desired. BO achieves this by (1) guiding the sampling with a prior belief about the objective, and (2) trading off exploration and exploitation through an acquisition function. Hence, it is necessary to specify a prior over functions and an acquisition function. Both choices are described briefly in the following. The explanations are largely based on [5].
Due to their convenient analytical properties, GPs (as introduced in Section 2.1) are a common choice as a prior $p(f)$ over functions. We therefore restrict ourselves to BO with a GP prior. Starting with such a prior belief about the function, Bayes' rule is used to maintain a posterior over possible objective functions f, i.e. $p(f|\mathscr{D}) \propto p(\mathscr{D}|f)p(f)$.
Additionally, an acquisition function is employed that takes into account the current belief about the objective to decide where to sample next. In case of a GP prior, the current knowledge about the function can be sufficiently described by the mean $\mu(\cdot)$ and variance $\sigma^2(\cdot)$ of the predictive distribution $p(f^*|\mathscr{D}, \mathbf{x}^*)$. Typically, the acquisition function encodes a trade-off between exploiting knowledge about regions with high function values and exploring the search space in areas with high uncertainty. That is because sufficient exploration is crucial to avoid premature convergence due to local optima. Hence, the next sample point is selected by maximizing the acquisition function $u(\cdot)$ given the data, i.e. $\mathbf{x} = \arg\max_x u(\mathbf{x}|\mathscr{D})$. The most popular acquisition functions in combination with a GP prior are introduced in the following.

PI PI [39] maximizes the probability of improving over the current highest function value $f(\mathbf{x}^*)$, where $\mathbf{x}^* = \arg\max_{x\in\mathscr{D}} f(x)$. The original version $\mathrm{PI}(\mathbf{x}) = p(f(\mathbf{x}) \geq f(\mathbf{x}^*))$ only focuses on exploitation as it favors points that probably result in higher values than $f(\mathbf{x}^+$ over sample points with high uncertainty. In order to encourage exploration, a user-defined trade-off parameter $\xi \geq 0$ can be introduced, yielding

$$\mathrm{PI}(\mathbf{x}) = p(f(\mathbf{x}) \geq f(\mathbf{x}^*) + \xi) = \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^*) - \xi}{\sigma(\mathbf{x})}\right),$$

where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. Higher ξ values lead to more exploration while $\xi = 0$ recovers the original formulation.
EI Instead of just looking for the probability of improvement, one could also consider the magnitude of the improvement. Because the true magnitude is unknown in advance, EI [40, 41] computes the expected deviation from the current best value $f(\mathbf{x}^*)$ under a Gaussian posterior distribution, which evaluates to

$$\mathrm{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - f(\mathbf{x}^* - \xi))\,\Phi(Z) + \sigma(\mathbf{x})\phi(Z),$$

where $Z = (\mu(\mathbf{x}) - f(\mathbf{x}^*) - \xi)/\sigma(\mathbf{x})$, $\phi(\cdot)$ is the standard normal density function, and $\mathrm{EI}(\mathbf{x}) = 0$ if $\sigma(\mathbf{x}) = 0$. The ξ parameter plays the same role as in PI. Empirically, $\xi = 0.01$ has been found to work well for many cases.
GP-UCB Motivated by continuous-armed bandits, GP-UCB [42] attempts to minimize regret during the optimization. In the basic multi-armed bandit formulation [43], for each round $t \geq 1$ one has to decide to play one of K discrete bandits, yielding rewards $R_{k,t} \in [0, 1]$, where $k = \{1, \ldots, K\}$ indexes the bandits. The rewards of bandit k are assumed to be i.i.d. random variables with unknown mean μ_k, and the rewards are independent across bandits. The goal is to find a policy π that selects the next bandit k before each rollout, such that the expected cumulative reward is maximized or, equivalently, the regret suffered from not picking the optimal point is minimized. The reward function therefore plays the role of the objective function $f(\cdot)$ in the BO framework, and the policy π is equivalent to the acquisition function $u(\cdot)$. Continuous-armed bandits generalize the original formulation to continuous inputs $\mathbf{x}$. In case a GP prior is employed, Srivanas et. al [42] propose the following acquisition function:

$$\text{GP-UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\beta}\sigma(\mathbf{x}), \tag{2.13}$$

where $\beta \geq 0$ is a trade-off parameter. They show that for certain choices of β it is possible to bound the regret.
A comparison of the different acquisition functions is provided in Fig. 2.2. As can be seen, PI and EI show similar behavior. In particular, they are more greedily searching in the vicinity of the current maximum compared to GP-UCB. The latter puts more weight on uncertain regions, even when they are far from the current maximum.

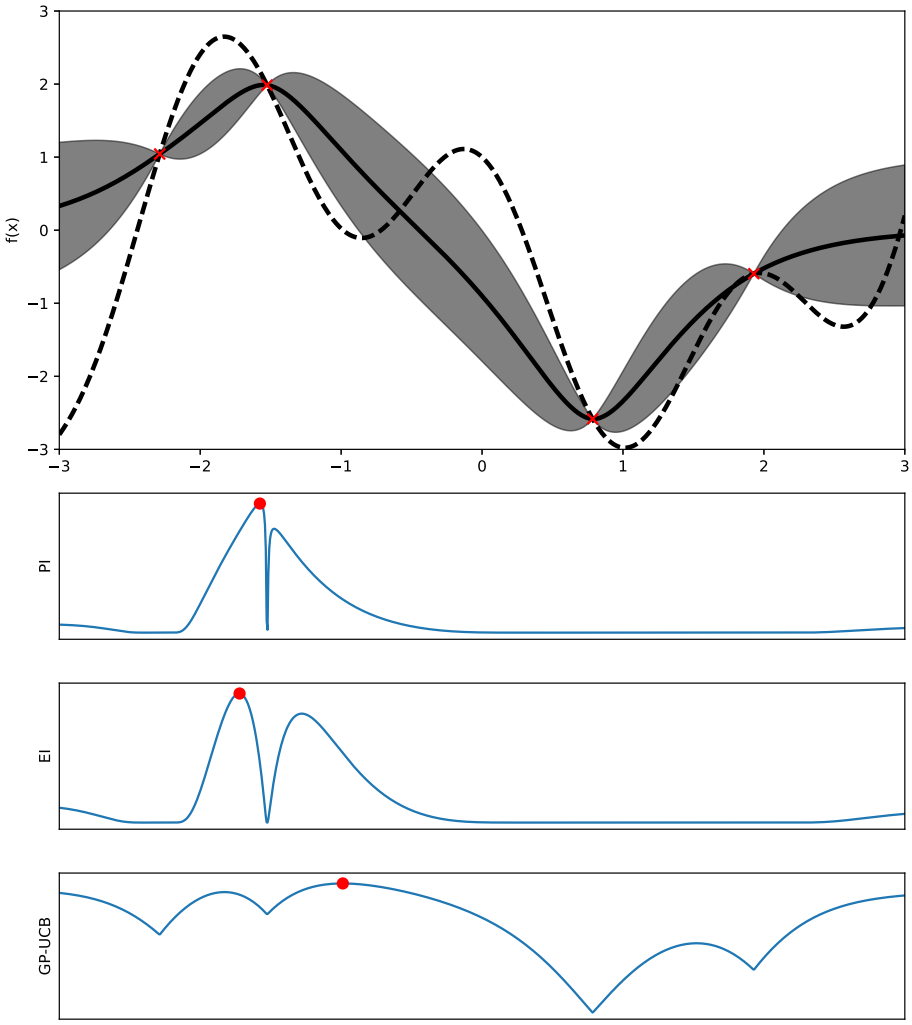


Figure 2.2: Comparison of different acquisition functions. At the top, the true function (dotted line) and the predictive posterior of a GP (solid line with error bars) are shown. Red crosses depict observations. Below, the PI, EI and GP-UCB acquisition functions are depicted. A red dot indicates the input location that maximizes the acquisition function. The figure is based on [5].

2.2.2 Active Learning

Active learning [44] is an area that is closely related to BO. It is commonly used to efficiently acquire data labels from an oracle to learn a model. In particular, active learning approaches decide when and for which candidates a label is required in order to improve the model. This property is useful in situations where querying the oracle is expensive or time-consuming, e.g. because labeling is performed by a human annotator.

Active learning can be used in many different settings. Most applications are concerned with the pool-based scenario, where the goal is to select the best sample from a pool of unlabeled data. A common approach is to score each label according to a measure of informativeness with respect to the model. The simplest strategy would be to query the sample about which the model is most uncertain about. For example, in case a GP model is used, one could choose the sample that maximizes the variance of the predictive posterior distribution.

A popular uncertainty measure from information theory is entropy. In general, the entropy of a random variable X with distribution p is defined as $H(X) = -\sum_i p(x_i) \log p(x_i)$. Of particular interest is the entropy of the posterior distribution of the model, i.e.

$$H = -\sum_i p(y_i|\mathbf{x}) \log p(y_i|\mathbf{x}),$$

where $\mathbf{x}$ are inputs and y_i are possible labels. For example, the entropy of a Gaussian distribution with covariance matrix $\mathbf{\Sigma}$ is given by

$$H(\mathcal{N}) = \frac{d}{2}\log(2\pi e) + \frac{1}{2}\log|\mathbf{\Sigma}|, \tag{2.14}$$

where d is the dimensionality of the data.
Another idea is to select the instance that would have the greatest impact on the model if the label was known. This strategy is also known as expected model change [44]. One way to quantify this change is to compare the predictive posterior distributions after and before the label is added to the dataset. The difference between two continuous distributions p and q can be measured by the KL divergence,

$$\mathrm{KL}(p||q) = \int p(\mathbf{x})\log\frac{p(\mathbf{x})}{p(\mathbf{y})},$$

which is also known as relative entropy or information gain. For instance, the KL between two multivariate Gaussian distributions $\mathcal{N}_1(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$ and $\mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$ is given by

$$\mathrm{KL}(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{2}\left(\mathrm{Tr}(\mathbf{\Sigma}_2^{-1}\mathbf{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\mathbf{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - d + \log\frac{|\mathbf{\Sigma}_2|}{|\mathbf{\Sigma}_1|}\right). \tag{2.15}$$

The KL divergence is always non-negative and non-symmetric. If the two distributions are the same, the KL is zero. Otherwise, the KL can be interpreted as the information that is gained by using the distribution p instead of q. From a Bayesian standpoint, p plays the role of the posterior distribution whereas q is the prior. The expected model change could therefore be expressed as the expectation of the KL between p and q over the possible labels, where q and p are the predictive distributions before and after adding the label respectively. An extensive overview of various other strategies is given in [44].
In contrast to pool-bases sampling, stream-based sampling is a scenario where instances are sampled sequentially, and the model decides whether a sample should be labeled or not. In this case, the uncertainty measures described above can be applied directly by introducing a threshold above which a label is requested.

2.3 Policy Search for Robotics

In a typical robotic RL setting [45], a robot operates in an environment by performing actions to achieve some goal. The state of the robot and the environment are jointly summarized as $\mathbf{x} \in \mathbf{X}$. Motor commands $\mathbf{u} \in \mathbf{U}$ are chosen by a control policy π and change the current state according to the transition function $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. In this thesis, we consider the episodic case where the task is limited to a given number of time steps T, resulting in rollouts or trajectories $\boldsymbol{\tau} = \{\mathbf{x}_1, \mathbf{u}_1, \dots \mathbf{x}_T, \mathbf{u}_T\}$. For each rollout, the robot receives reward $R(\boldsymbol{\tau})$. The goal is to find an optimal policy π^* that maximizes the expected reward,

$$J_\pi = \mathbb{E}[R(\boldsymbol{\tau})|\pi] = \int p_\pi(\boldsymbol{\tau})R(\boldsymbol{\tau})d\boldsymbol{\tau},$$

where $p_\pi(\boldsymbol{\tau})$ denotes the distribution over trajectories under policy π. The reward $R(\boldsymbol{\tau})$ often consists of a sum of instantaneous rewards $r(\mathbf{x}_t, \mathbf{u}_t)$, i.e. $R(\boldsymbol{\tau}) = \sum_{t=0}^{T} r(\mathbf{x}_t, \mathbf{u}_t)$. However, for some tasks it is common to only provide a final reward evaluating task achievement. For instance, consider the task of grasping an object, where the states $\mathbf{x}$ are defined by the hand configuration of the robot as well as the position and orientation of the object. The goal of the robot is to lift the object by applying suitable torques $\mathbf{u}$. A positive reward is given if the object was lifted successfully, and a negative reward otherwise. This task, like many others in the robotics domain, consists of a high-dimensional, continuous state-action space. Policy search (PS) methods are well suited for these kind of settings. We use an extended version of the standard PS framework that is introduced in the following.

2.3.1 Contextual Policy Search

In order to facilitate optimization in high-dimensional state-action spaces, PS methods [46] use a pre-structured lower-level control policy $\pi(\mathbf{x}, \boldsymbol{\omega})$ parametrized by $\boldsymbol{\omega} \in \mathbf{\Omega}$. This lower-level policy is usually deterministic, $\mathbf{u} = \pi(\mathbf{x}, \boldsymbol{\omega})$, and could for example be a dynamic movement primitive [47] or linear feedback controller. The policy parameters $\boldsymbol{\omega}$ are provided by an upper-level policy $\pi(\boldsymbol{\omega})$, which often takes the form of a Gaussian, i.e. $\pi(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega)$. PS methods directly search for the optimal upper-level policy

$$\pi^*(\boldsymbol{\omega}) = \arg\max_\pi \int \pi(\boldsymbol{\omega})p(\boldsymbol{\tau}|\boldsymbol{\omega})R(\boldsymbol{\tau})d\boldsymbol{\tau}d\boldsymbol{\omega},$$

thereby reducing the search space considerably. One shortcoming of the standard PS formulation is that it does not generalize well to multiple contexts $\mathbf{s} \sim \mu(\mathbf{s})$, where a context completely specifies the task and is drawn from an unknown, task-dependent distribution $\mu(\mathbf{s})$ before the rollout. For instance, a context could be a contact location specifying where to grasp an object or the target position for throwing a ball. In the original PS framework it would be necessary to learn a separate policy $\pi_s(\boldsymbol{\omega})$ for each context $\mathbf{s}$, which quickly becomes infeasible for continuous context variables. Contextual PS [48, 49] addresses this issue by searching for an upper-level policy $\pi(\boldsymbol{\omega}|\mathbf{s})$ given context $\mathbf{s}$,

$$\pi^*(\boldsymbol{\omega}|\mathbf{s}) = \arg\max_\pi \int \mu(\mathbf{s})\pi(\boldsymbol{\omega}|\mathbf{s})R_{\mathbf{s}\boldsymbol{\omega}} d\boldsymbol{\omega} d\mathbf{s}, \tag{2.16}$$

where $R_{\mathbf{s}\boldsymbol{\omega}} = \int p(\boldsymbol{\tau}|\mathbf{s},\boldsymbol{\omega})R(\boldsymbol{\tau},\mathbf{s})d\boldsymbol{\tau}$ denotes the expected reward of the trajectory given context $\mathbf{s}$ and parameter $\boldsymbol{\omega}$. In this thesis, contextual relative entropy policy search (REPS) [46, 50] is used, which is a particular contextual PS method motivated by information-theoretic considerations. Contextual REPS is now explained in more detail.

2.3.2 Contextual Relative Entropy Policy Search

Information-theoretic approaches like REPS [51] attempt to stay close to the data while optimizing the policy. In the episodic case, this means that the trajectory distribution induced by the new policy should not be too far away from the trajectory distribution that was observed before the policy update. From a robotics perspective, such a property is particularly desirable as large jumps in the state space might be unsafe for robot operation. Because the context $\mathbf{s}$ and parameter $\boldsymbol{\omega}$ uniquely determine the trajectory distribution [46], it is possible to treat the entire optimization problem with only the joint distribution $p(\mathbf{s},\boldsymbol{\omega}) = p(\mathbf{s})\pi(\boldsymbol{\omega}|\mathbf{s})$. Contextual REPS [46, 50] therefore aims to maximize the expected reward

$$J_{\mathbf{s}\boldsymbol{\omega}} = \mathbb{E}[R_{\mathbf{s}\boldsymbol{\omega}}] = \int p(\mathbf{s},\boldsymbol{\omega})R_{\mathbf{s}\boldsymbol{\omega}} d\boldsymbol{\omega} d\mathbf{s} \tag{2.17}$$

with respect to $p(\mathbf{s},\boldsymbol{\omega})$, cf. Eq. 2.3. The optimization is subject to bounding the relative entropy or Kullback-Leibler divergence (KL) between the context-parameter distribution $p(\mathbf{s},\boldsymbol{\omega})$ and the observed joint distribution $q(\mathbf{s},\boldsymbol{\omega})$,

$$\mathrm{KL}(p(\mathbf{s},\boldsymbol{\omega}))||q(\mathbf{s},\boldsymbol{\omega})) = \int p(\mathbf{s},\boldsymbol{\omega}) \log \frac{p(\mathbf{s},\boldsymbol{\omega})}{q(\mathbf{s},\boldsymbol{\omega})} d\boldsymbol{\omega} d\mathbf{s} \leq \epsilon, \tag{2.18}$$

where $\epsilon \geq 0$ is the desired KL bound. This constraint effectively allows to limit the information loss due to the policy update, thereby trading off exploration and exploitation. The larger ϵ, the more aggressive the policy is updated. However, the context distribution $p(\mathbf{s}) = \int p(\mathbf{s},\boldsymbol{\omega})d\boldsymbol{\omega}$ cannot be freely chosen since it is defined by the learning problem as $\mu(\mathbf{s})$. Therefore, the estimated context distribution $p(\mathbf{s})$ needs to always equal the true distribution, i.e. $\forall \mathbf{s} : p(\mathbf{s}) = \mu(\mathbf{s})$. To avoid having to add infinitely many constraints in case of continuous contexts $\mathbf{s}$, we only match feature expectations instead, i.e.

$$\int p(\mathbf{s})\boldsymbol{\phi}(\mathbf{s}) d\mathbf{s} = \hat{\boldsymbol{\phi}}, \tag{2.19}$$

where $\boldsymbol{\phi}(\mathbf{s})$ are features of context $\mathbf{s}$, and $\hat{\boldsymbol{\phi}} = \int \mu(\mathbf{s})\boldsymbol{\phi}(\mathbf{s})d\mathbf{s}$ are the average observed context features. Lastly, we need to make sure that $p(\mathbf{s},\boldsymbol{\omega}) = p(\mathbf{s})\pi(\boldsymbol{\omega}|\mathbf{s})$ defines a probability distribution,

$$\int p(\mathbf{s},\boldsymbol{\omega}) d\boldsymbol{\omega} d\mathbf{s} = 1. \tag{2.20}$$

Combining the optimization objective from Eq. 2.17 with the constraints in Eq. 2.18–2.20 yields the following constrained optimization problem:

$$\begin{aligned}
\max_p & \int p(\mathbf{s},\boldsymbol{\omega})R_{\mathbf{s}\boldsymbol{\omega}} d\boldsymbol{\omega} d\mathbf{s} \\
\text{s.t.} & \int p(\mathbf{s},\boldsymbol{\omega}) \log \frac{p(\mathbf{s},\boldsymbol{\omega})}{q(\mathbf{s},\boldsymbol{\omega})} d\boldsymbol{\omega} d\mathbf{s} \leq \epsilon, \\
& \int p(\mathbf{s},\boldsymbol{\omega})\boldsymbol{\phi}(\mathbf{s}) d\boldsymbol{\omega} d\mathbf{s} = \hat{\boldsymbol{\phi}}, \\
& \int p(\mathbf{s},\boldsymbol{\omega}) d\boldsymbol{\omega} d\mathbf{s} = 1.
\end{aligned}$$

As shown in the original work [50, 51], it is possible to obtain a closed-form solution for this optimization problem using the method of Lagrangian multipliers,

$$p(\mathbf{s}, \boldsymbol{\omega}) \propto q(\boldsymbol{\omega}|\mathbf{s})\mu(\mathbf{s}) \exp\left(\frac{R_{\mathbf{s}\boldsymbol{\omega}} - V(\mathbf{s})}{\eta}\right), \tag{2.21}$$

where $V(\mathbf{s}) = \boldsymbol{\phi}(\mathbf{s})^T \boldsymbol{\nu}$ is a context-dependent baseline that can be interpreted as a value function [51]. The parameters η and $\boldsymbol{\nu}$ are Lagrangian multipliers, which are obtained by minimizing the dual function,

$$g(\boldsymbol{\nu}, \eta) = \epsilon\eta + \boldsymbol{\nu}^T \hat{\boldsymbol{\phi}} + \eta \log \int q(\mathbf{s}, \boldsymbol{\omega}) \exp\left(\frac{R_{\mathbf{s}\boldsymbol{\omega}} - V(\mathbf{s})}{\eta}\right) d\boldsymbol{\omega} d\mathbf{s} \tag{2.22}$$

subject to $\eta > 0$. In practice, Eq. 2.22 is approximated using samples $\mathcal{D} = \{(\mathbf{s}_i, \boldsymbol{\omega}_i, R_i)\}_{i=1,\dots,N}$ collected from rollouts, turning the integrals into sums. Each sample is weighted by probability d_i,

$$d_i \propto \exp((R_i - V(\mathbf{s}_i))/\eta), \tag{2.23}$$

where the context-parameter distribution $q(\mathbf{s}, \boldsymbol{\omega})$ has been omitted (cf. Eq. 2.21) since we already sampled from this distribution. Note that the expected reward $R_{\mathbf{s}\boldsymbol{\omega}}$ is only approximated by a single reward sample R, which can be troublesome in case of high reward variance. That is because the exponential weighting of the reward can favor risk-seeking policies [50].

The algorithm proceeds as follows. First, the policy is initialized, e.g. through random rollouts. Then, a context $\mathbf{s}_i \sim \mu(\mathbf{s})$ is observed prior to each rollout, which is used to determine the lower-level parameters $\boldsymbol{\omega}_i \sim \pi(\boldsymbol{\omega}|\mathbf{s}_i)$. The rollout is performed according to $\boldsymbol{\omega}_i$, yielding reward R_i. After N rollouts, the policy is updated. This process is repeated until convergence.

3 Hierarchical Reinforcement Learning with Dual Reward Estimation from Preferences

Our goal is to learn and select between multiple options k, where each option is associated with a lower-level policy π_l^k, and the reward function is assumed to be unknown. Moreover, it is assumed that the correct option to choose depends on the current context of the scene. In particular, we are interested in learning multiple grasp types that allow to perform versatile grasping motions depending on the object.
We propose a hierarchical reinforcement learning framework consisting of an upper-level selection policy and multiple lower-level action policies corresponding to the different options. Furthermore, two different reward models are learned: (1) an outcome reward model that evaluates the outcomes of rollouts based on human preference feedback, and (2) a context-parameter reward-model per option that predicts the context and option to choose for the next rollout. Data-efficiency is achieved through BO and active learning techniques. The complete approach is outlined in Algorithm 1 and described in more detail in the following.

Algorithm 1 Hierarchical reinforcement learning with dual reward estimation from preferences

Input: number of options K, active learning threshold λ, number of rollouts per epsiode N
Initialization: initialize reward models $p(R_o|\mathscr{D}_\succ, \mathbf{o})$, $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$, and policies π_l^k randomly or from demonstrations

repeat
  Observe global context $\tilde{\mathbf{s}} \sim \mu(\tilde{\mathbf{s}})$
  for $k = 1$ **to** K **do**
    for all $\mathbf{s}_i^k \in \mathscr{S}^k(\tilde{\mathbf{s}})$ **do**
      Approximate $\mathbb{E}[R_{s\omega}]$ and $\mathbb{V}[R_{s\omega}]$ with samples $R_{s\omega} \sim p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}_i^k, \boldsymbol{\omega}_j)$, $\boldsymbol{\omega}_j \sim \pi_l^k(\boldsymbol{\omega}|\mathbf{s}_i^k)$
    end for
  end for
  Choose $\mathbf{s}^*, k^*$ according to π_u
  Perform rollout $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}^*, \boldsymbol{\omega})$ with $\boldsymbol{\omega} \sim \pi_l^k(\boldsymbol{\omega}|\mathbf{s}_i^k)$
  Observe outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$
  if expected information gain from preference feedback for $p(R_o|\mathscr{D}_\succ, \mathbf{o}) > \lambda$ **then**
    Query human for preference feedback
    Update outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$
  end if
  if episode completed **then**
    Update outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$
    Update context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$
    Update lower-level policies π_l^k
  end if
until task learned
return learned policies π_l^k

Prior to each rollout, a global context $\tilde{\mathbf{s}} \sim \mu(\tilde{\mathbf{s}})$ is observed that specifies the task and is assumed to be continuous and possibly high-dimensional. The task can be solved using one of K options. For instance, in case of grasping the global context $\tilde{\mathbf{s}}$ could be the point cloud of an object to be grasped, and the options are different grasp types. However, because of the high dimensionality of such a global context, we usually only work with feature vectors $\mathbf{s} = \phi_s(\tilde{\mathbf{s}})$ that succinctly describe the global context. In case of grasping, a feature vector could for example be the contact location for grasping an object. In our framework, we generate multiple local contexts $\mathscr{S}^k = \{\mathbf{s}_i^k = \phi_s^k(\tilde{\mathbf{s}})\}_{i=1,\dots,n}$ for each option k and estimate the expected reward $\mathbb{E}[R_{s\omega}]$ of trajectories produced by using lower-level policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$ with context $\mathbf{s}_i^k$. In particular, we approximate the expected reward by samples $R_{s\omega} \sim p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}_i^k, \boldsymbol{\omega}_j)$ from a probabilistic context-parameter reward model for each option k, where policy parameters $\boldsymbol{\omega}_j \sim \pi_l^k(\boldsymbol{\omega}|\mathbf{s}_i^k)$ are generated from lower-level policy π_l^k. Given the expected rewards, we select the best context-option pair $(\mathbf{s}^*, k^*)$ according to an upper-level policy π_u.
Once an option k^* has been selected, a rollout is performed using policy parameters $\boldsymbol{\omega} \sim \pi_l^{k^*}(\boldsymbol{\omega}|\mathbf{s}^*)$. After the rollout, the trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}^*, \boldsymbol{\omega})$ needs to be evaluated. A convenient choice is to use the estimated reward $R_{s\omega}$. However,

system noise and uncertainty about the inputs can cause very different outcomes than assumed during planning time. Instead, we propose to learn a separate reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ based on the outcomes of rollouts, where an outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$ is a succinct description of trajectory $\boldsymbol{\tau}$. The outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ is learned from human preferences and updated after N rollouts. To reduce the number of feedback requests, we only query the human if the expected information gain from a preference relation is large enough. Based on this model, the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ and policies π_l^k are updated.
The remainder of this chapter is organized as follows. Section 3.1 describes how the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ and the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ are learned. In Section 3.2, the upper-level policy π_u as well as the policy learning approach for obtaining lower-level policies π_l^k are explained in more detail. Finally, the proposed framework is tailored towards a robotic grasping task in Section 3.3.

3.1 Dual Reward Estimation from Preferences

In order to select a context-option pair $(\mathbf{s}_i^k, k)$, we need to evaluate the expected reward $\mathbb{E}[R_{s\omega}]$ of using lower-level policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s}_i^k)$ with context $\mathbf{s}_i^k$ of option k. Since the true reward function is unknown, we learn a context-parameter reward model $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ for each option k. Learning multiple reward models allows to deal with the multi-modal nature of the joint data $\bigcup_{k=1}^K \mathscr{D}^k$ by treating each mode of the data distribution independently.
Once a rollout has been performed, the trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}_i^k, \boldsymbol{\omega})$ has to be evaluated. However, while the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ are necessary for predicting which context-option pair to choose, there are better ways for estimating the reward of the actual trajectory. That is because system noise and uncertainty about policy parameter $\boldsymbol{\omega}$ can lead to very different trajectories than assumed before the rollout. Instead, we learn a separate outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ based on outcomes $\mathbf{o}$, where an outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$ is a low-dimensional feature vector describing the trajectory. This model is learned from human preferences about outcomes across all options. Since an outcome $\mathbf{o}$ usually is of lower dimensionality than a context-parameter pair $(\mathbf{s}, \boldsymbol{\omega})$, the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ can be learned faster than the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$. Given the outcome reward model, it is then possible to associate the estimated outcome rewards with the context-parameter pairs $(\mathbf{s}, \boldsymbol{\omega})$ that led to the outcomes. Finally, the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ can be learned using the augmented datasets $\mathscr{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}^*(\mathbf{o}_i)\}_{i=1,\dots,N_k}$, where $\bar{R}^*(\mathbf{o})$ is the predicted mean reward from outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$.
Intuitively, the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ attempts to model the reward function of the human. When a human evaluates a rollout, it judges the quality of the outcome of the rollout, not the internal parameters of the robot. On the other hand, the robot needs to evaluate the quality of certain context-parameter pairs $(\mathbf{s}, \boldsymbol{\omega})$ through the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$.
The context-parameter reward model $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ is described in more detail in Section 3.1.1. Subsequently, the outcome reward model is explained in Section 3.1.2.

3.1.1 Parameter Reward Model

We learn a probabilistic reward model $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ for each policy π_l^k from dataset $\mathscr{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}^*(\mathbf{o}_i)\}_{i=1:N_k}$, where $\mathbf{o} = \phi(\boldsymbol{\tau})$ is the outcome of trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}, \boldsymbol{\omega})$, and $\bar{R}^*(\mathbf{o})$ is the predicted mean reward using outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$. The outcome reward model is introduced later in Section 3.1.2. For now, it is sufficient to assume that the reward $R(\mathbf{x})$ for the context-parameter pair $\mathbf{x} = [\mathbf{s}, \boldsymbol{\omega}]^T$ is given by the mean reward for the outcome induced by $\mathbf{x}$, i.e. $R(\mathbf{x}) := \bar{R}^*(\mathbf{o})$. We additionally assume that the predicted rewards are only noisy versions of the true rewards, $R(\mathbf{x}) = R_{s\omega}(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is i.i.d. Gaussian noise. We use GP regression (see Section 2.1.1) to model the reward function, i.e.

$$R_{s\omega}(\mathbf{x}) \sim \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{3.1}$$

where the mean function is assumed to be zero, $m(\mathbf{x}) = \mathbf{0}$, and the covariance function $k(\mathbf{x}, \mathbf{x}')$ is given by the squared exponential covariance function from Eq. 2.2. The hyperparameters $\boldsymbol{\theta}$ are estimated by maximizing the marginal likelihood. The predictive distribution $p(R_{s\omega}^*|\mathbf{R}) = \mathcal{N}(\mu_*, \sigma_*^2)$ at a test point $\mathbf{x}^*$ is Gaussian with

$$\mu_* = \boldsymbol{\kappa}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{R}, \tag{3.2a}$$

$$\sigma_*^2 = \kappa - \boldsymbol{\kappa}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\boldsymbol{\kappa}. \tag{3.2b}$$

where $\mathbf{R} = [R(\mathbf{x}_1), \dots, R(\mathbf{x}_{N_k})]$, $\boldsymbol{\kappa} = [k(\mathbf{x}_1, \mathbf{x}^*), \dots, k(\mathbf{x}_{N_k}, \mathbf{x}^*)]^T$, $\kappa = k(\mathbf{x}^*, \mathbf{x}^*)$, and $\mathbf{I}$ is the identity matrix. Given the predictive distribution, it is possible to predict the reward $R_{s\omega}$ for each context-parameter pair $(\mathbf{s}, \boldsymbol{\omega})$. Based on those estimates, the goal is to select the best possible context and option for the next rollout. The sample mean μ_R and variance σ_R^2 are taken into consideration by the upper-level policy π_u (introduced in Section 3.2.1) to decide which context-option

pair to choose. Once a rollout has been performed with a particular context and option, it needs to be evaluated. In the next section, we introduce the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ that allows to predict the reward of an outcome $\mathbf{o}$ of a rollout.

3.1.2 Outcome Reward Model

After each rollout, the quality of the actual trajectory $\boldsymbol{\tau}$ has to be estimated again. We propose to use a separate reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ based on the outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$. This model is learned from human preferences.
Let $\mathscr{O} = \{\mathbf{o}_1, \dots, \mathbf{o}_N\}$ with $N = \sum_{k=1}^{K} N_k$ be the outcomes of rollouts across all options k, and $\mathscr{D}_\succ = \{\mathbf{v}_1 \succ \mathbf{u}_1, \dots, \mathbf{v}_M \succ \mathbf{u}_M\}$ with $\mathbf{v}_m \in \mathscr{O}, \mathbf{u}_m \in \mathscr{O}$ are M preference relations over those outcomes. The outcomes are assumed to be valued according to a noisy utility function $y(\cdot) = R_o(\cdot) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is i.i.d. Gaussian noise. Using the probabilistic GP preference learning model as introduced in Section 2.1.2, the latent rewards can be modeled as

$$R_o(\mathbf{o}) \sim \mathrm{GP}(m(\mathbf{o}), k(\mathbf{o}, \mathbf{o}')), \tag{3.3}$$

where the mean function $m(\cdot)$ is set to zero, and the squared exponential covariance function $k(\cdot,\cdot)$ is employed (Eq. 2.2). After performing a Laplace approximation, the predictive distribution $p(\mathbf{R}_o^*|\mathscr{D}_\succ)$ for two test instances $\mathbf{r} \in \mathscr{O}, \mathbf{s} \in \mathscr{O}$ with $\mathbf{R}_o^* = [R_o(\mathbf{r}), R_o(\mathbf{s}]^T$ is given by a Gaussian $\mathcal{N}(\bar{\mathbf{R}}^*, \boldsymbol{\Sigma}^*)$ with mean $\bar{\mathbf{R}}^*$ and covariance $\boldsymbol{\Sigma}^*$,

$$\bar{\mathbf{R}}^* = [\mu_r^*, \mu_s^*] = \boldsymbol{\kappa}^T \mathbf{K}^{-1} \mathbf{R}_{\mathrm{MAP}}, \tag{3.4}$$

$$\boldsymbol{\Sigma}^* = \begin{bmatrix} \Sigma_{rr}^* & \Sigma_{rs}^* \\ \Sigma_{sr}^* & \Sigma_{ss}^* \end{bmatrix} = \mathbf{K}_* - \boldsymbol{\kappa}^T (\mathbf{K} + \boldsymbol{\Gamma}_{\mathrm{MAP}}^{-1})^{-1} \boldsymbol{\kappa}, \tag{3.5}$$

where $\boldsymbol{\kappa} = \begin{bmatrix} k(\mathbf{r}, \mathbf{o}_1), \dots, k(\mathbf{r}, \mathbf{o}_N) \\ k(\mathbf{s}, \mathbf{o}_1), \dots, k(\mathbf{s}, \mathbf{o}_N) \end{bmatrix}^T$, $\mathbf{K}_* = \begin{bmatrix} k(\mathbf{r}, \mathbf{r}) & k(\mathbf{r}, \mathbf{s}) \\ k(\mathbf{s}, \mathbf{r}) & k(\mathbf{s}, \mathbf{s}) \end{bmatrix}$, $\mathbf{R}_{\mathrm{MAP}} = [R_o(\mathbf{o}_1), \dots, R_o(\mathbf{o}_N)]\big|_{\mathbf{R}_o = \mathbf{R}_{\mathrm{MAP}}}$ are the estimated latent rewards of the MAP solution, and $\boldsymbol{\Gamma}_{\mathrm{MAP}} = \nabla^2 - \log p(\mathscr{D}_\succ|\mathbf{R}_o)\big|_{\mathbf{R}_o = \mathbf{R}_{\mathrm{MAP}}}$ is the Hessian of the negative log likelihood at the MAP estimate. The predictive preference $p(\mathbf{r} \succ \mathbf{s}|\mathscr{D}_\succ)$ can by obtained by

$$p(\mathbf{r} \succ \mathbf{s}|\mathscr{D}_\succ) = \Phi\left(\frac{\mu_r^* - \mu_s^*}{\sigma_*}\right), \tag{3.6}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function, and $\sigma_*^2 = 2\sigma_n^2 + \Sigma_{rr}^* + \Sigma_{ss}^* - \Sigma_{rs}^* - \Sigma_{sr}^*$.
We use the predictive distribution $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ to estimate the reward of outcome $\mathbf{o}$, which is added to the dataset $\mathscr{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}(\mathbf{o}_i)\}_{i=1,\dots,N_k}$ of the corresponding option k.
Furthermore, the predictive preference $p(\mathbf{q} \succ \mathbf{r}|\mathscr{D}_\succ)$ can be used to decide whether it is necessary to query the human for preference feedback. Such an active learning strategy is described in the following.

Active Learning

Since repeatedly giving preference feedback is a tedious task for humans, it is desirable to minimize the number of feedback requests. In such cases, active learning techniques (see Section 2.2.2) can be used. As instances arrive sequentially, stream-based approaches are of particular interest.
Hence, we only query the human if the expected information gained from a preference relation is larger than some user-defined threshold λ. The information gain of including a preference relation $\mathbf{q} \succ \mathbf{r}$ can be expressed as the KL (Eq. 2.15) between the posterior after and before adding the preference relation, $\mathrm{KL}(p(\mathbf{R}|\mathscr{D}_{\mathbf{q}\succ\mathbf{r}})||p(\mathbf{R}|\mathscr{D}))$, where $\mathscr{D}_{\mathbf{q}\succ\mathbf{r}} = \mathscr{D} \cup \{\mathbf{q} \succ \mathbf{r}\}$ is an updated dataset that includes the new preference relation. Hence, we define the following active learning criterion,

$$p(\mathbf{q} \succ \mathbf{r}|\mathscr{D})\mathrm{KL}(p(\mathbf{R}|\mathscr{D}_{\mathbf{q}\succ\mathbf{r}})||p(\mathbf{R}|\mathscr{D})) + p(\mathbf{r} \succ \mathbf{q}|\mathscr{D})\mathrm{KL}(p(\mathbf{R}|\mathscr{D}_{\mathbf{r}\succ\mathbf{q}})||p(\mathbf{R}|\mathscr{D})) > \lambda, \tag{3.7}$$

where the information gain is weighted by the predictive preference. The threshold parameter λ allows to trade off accuracy and data-efficiency, i.e. higher λ values lead to less feedback requests whereas lower λ values prompt many judgements that improve the reward model. A similar criterion for preference learning GPs based on entropy was proposed by Chu and Ghahramani [52]

3.2 Hierarchical Reinforcement Learning

This section describes the policies used in the hierarchical framework. In total, there are three types of policies: (1) the upper-level policy π_u decides which context-option pair should be selected, (2) the lower-level policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s}_i^k)$ generates policy parameter $\boldsymbol{\omega}$ given context $\mathbf{s}_i^k$ of option k, and (3) the sub-policy $\pi(\mathbf{x}, \boldsymbol{\omega})$ generates actions $\mathbf{u}$ according to state $\mathbf{x}$ and policy parameter $\boldsymbol{\omega}$. For instance, grasping an object requires to choose a contact location and a grasp type depending on the object to grasp. Once a context-option pair has been selected, motion parameters are generated by a lower-level policy, which govern a deterministic sub-policy that finally generates a grasping trajectory.
In Section 3.2.1, the upper-level policy is explained whereas Section 3.2.2 covers the lower-level policies in more detail. The deterministic sub-policy is task-dependent and will be specified later.

3.2.1 Upper-Level Policy

Prior to each rollout, the robot has to decide which context-option pair $(\mathbf{s}, k)$ to choose in order to maximize the cumulative reward. For instance, in robotic grasping we are interested in choosing a certain grasp type and location given the estimated quality of the resulting grasp. The selection is performed by upper-level policy π_u that has access to the predictive mean rewards $\mathbb{E}[R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega}] = \mu_R(\mathbf{s}_i^k, \boldsymbol{\omega})$ and their variances $\mathbb{V}[R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega}] = \sigma_R^2(\mathbf{s}_i^k, \boldsymbol{\omega})$. A simple policy would select the context-option pair with the highest expected reward under policy π_l^k, i.e.

$$(\mathbf{s}^*, k^*) = \arg\max_{\mathbf{s},k} \mathbb{E}_{\pi_l^k}\left[\mathbb{E}[R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega}]\right] \tag{3.8}$$

However, this policy would not make full use of the available information. Having a probabilistic model of the reward function at hand, it is possible to use BO techniques as a way to trade off exploration and exploitation. In particular, we make use of the GP-UCB acquisition function (Eq. 2.13). But in contrast to the standard continuous-armed bandit problem that GP-UCB addresses, we additionally observe a context $\mathbf{s}$. This setting is also known as a contextual continuous-armed bandit problem. The GP-UCB acquisition function can be extended straightforwardly to the contextual case [53],

$$\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega}) = \mu_R(\mathbf{s}, \boldsymbol{\omega}) + \sqrt{\beta}\sigma_R(\mathbf{s}, \boldsymbol{\omega}), \tag{3.9}$$

where $\beta > 0$ is a trade-off parameter. To account for the uncertainty about the policy parameters $\boldsymbol{\omega}$, the expectation of the acquisition function with respect to the policy needs to be considered, yielding the following upper-level policy π_u:

$$\pi_u : (\mathbf{s}^*, k^*) = \arg\max_{\mathbf{s},k} \mathbb{E}\left[\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega})\right], \tag{3.10}$$

where we approximate the expectation with samples $\boldsymbol{\omega}_j \sim \pi_l^k(\boldsymbol{\omega}|\mathbf{s})$, i.e.

$$\mathbb{E}\left[\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega})\right] \approx \frac{1}{M}\sum_{j=1}^{M} \text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega}_j). \tag{3.11}$$

Lastly, the trade-off parameter β has to be set. While there exist theoretical results that suggest certain β values [53], we deviate from them for two reasons. First, in contrast to the standard setting, the reward distribution of our approach is non-stationary as the lower-level policies π_l^k are changed through policy updates. Second, it is desirable to encourage equal exploration across the K options, in particular in the beginning. Inspired by the confidence bound from the UCB1 approach [43], we empirically found that $\beta = 2\log(N)/N_k$ works well to achieve this kind of behavior, where N_k are the number of rollouts with option k, and $N = \sum_{k=1}^{K} N_k$ is the total amount of rollouts.

3.2.2 Lower-Level Policy

Our goal is to learn K lower-level policies $\pi_l^k(\mathbf{s}, \boldsymbol{\omega})$, where each policy maximizes the expected long-term reward given the observed data. Since each policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$ depends on the current context $\mathbf{s}$, we employ contextual REPS (see Section 2.3.2) to learn the policies. In the contextual PS framework, $\pi(\boldsymbol{\omega}|\mathbf{s})$ is called the upper-level policy whereas $u = \pi(\mathbf{x}, \boldsymbol{\omega})$ denotes the lower-level policy. Because in our framework $\pi(\boldsymbol{\omega}|\mathbf{s})$ are lower-level policies themselves, we refer to $\pi(\boldsymbol{\omega}|\mathbf{s})$ and $\pi(\mathbf{x}, \boldsymbol{\omega})$ as lower-level policy and sub-policy, respectively.
Every lower-level policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$ corresponds to an option k with dataset $\mathscr{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}(\mathbf{o}_i)\}_{i=1,\ldots,N_k}$, where $\mathbf{o} = \phi(\boldsymbol{\tau})$ is the outcome of trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}, \boldsymbol{\omega})$, and $\bar{R}(\mathbf{o})$ is the predicted mean reward for outcome $\mathbf{o}$. The sample rewards

are converted into weights d_i using an exponential transformation (Eq. 2.23). The lower-level policies are assumed to be Gaussian, $\pi_l^k(\boldsymbol{\omega}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$, with mean $\boldsymbol{\mu}_\omega$ and covariance $\boldsymbol{\Sigma}_\omega$ [46],

$$\boldsymbol{\mu}_\omega = \boldsymbol{\phi}(\mathbf{s})^T \mathbf{w}, \tag{3.12a}$$

$$\boldsymbol{\Sigma}_\omega = \frac{\sum_{i=1}^N d_i \left(\boldsymbol{\theta}_\mathbf{i} - \mathbf{w}^T \phi(\mathbf{s}_\mathbf{i})\right)\left(\boldsymbol{\theta}_\mathbf{i} - \mathbf{w}^T \phi(\mathbf{s}_\mathbf{i})\right)^T}{1 - \sum_{i=1}^N d_i^2}, \tag{3.12b}$$

where $\mathbf{w}$ are linear weights on the context features $\phi(\mathbf{s})$. We use the squared exponential function for defining the context weights $\phi(\mathbf{s}) \in \mathcal{R}^M$ using M random context samples $\mathbf{s}_m$, i.e. the m-th element is defined as $\phi_m(\mathbf{s}) = k(\mathbf{s}, \mathbf{s}_m)$, where $k(\cdot,\cdot)$ is given by Eq. 2.2 with only a scalar bandwidth parameter ℓ. The hyperparameters $\boldsymbol{\theta} = \{\sigma_f, \ell\}$ are to be specified by the user. The weights $\mathbf{w}$ are found by unbiased weighted maximum likelihood estimation using N samples,

$$\mathbf{w}^* = \arg\max_\mathbf{w} \sum_{i=1}^N d_i \pi_\mathbf{w}(\boldsymbol{\omega}|\mathbf{s}) \tag{3.13}$$

$$= (\boldsymbol{\Phi}^T \mathbf{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{D} \boldsymbol{\Omega}, \tag{3.14}$$

where $\boldsymbol{\Phi} = [\phi_1, \ldots, \phi_N]$ and $\boldsymbol{\Omega} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_N]$ contain the feature vectors and policy parameters for each sample respectively, and $\mathbf{D}$ is a diagonal weighting matrix containing the weights d_i.
A potential pitfall of this approach is that the sample covariance $\boldsymbol{\Sigma}_\omega$ is often estimated from only few samples. In such cases, the weighted maximum likelihood estimate of the covariance matrix is prone to have high variance, in particular when dealing with high dimensionality. As a result, the policy search algorithm might convergence prematurely. In the spirit of covariance shrinkage methods, the covariance estimation with controlled entropy reduction (CECER) [54] method attempts to avoid this effect by weighting the new covariance estimate $\boldsymbol{\Sigma}_\omega$ with the old covariance matrix $\boldsymbol{\Sigma}_q$ of the sampling distribution q,

$$\hat{\boldsymbol{\Sigma}}_\omega = \lambda \boldsymbol{\Sigma}_q + (1-\lambda)\boldsymbol{\Sigma}_\omega, \tag{3.15}$$

where $\lambda \in [0,1]$ is a tradeoff parameter. CECER chooses λ such that the entropy of the Gaussian policy (Eq. 2.14) is reduced by a desired amount ΔH,

$$H(\boldsymbol{\Sigma}_q) - H(\lambda \boldsymbol{\Sigma}_q + (1-\lambda)\boldsymbol{\Sigma}_\omega) = \Delta H, \tag{3.16}$$

where $\Delta H = \alpha N_{\text{eff}}$, $N_{\text{eff}} = 1/\sum_i (d_i)^2$ is the number of effective samples that were used to compute $\boldsymbol{\Sigma}_\omega$, and α is a hyperparameter. Intuitively, more effective samples allow a higher entropy reduction, which leads to a lower λ value as more weight is put on the new sample covariance matrix $\boldsymbol{\Sigma}_\omega$. The correct λ value can be found through an exhaustive search. [54]

3.3 Hierarchical Reinforcement Learning with Dual Reward Estimation for Robotic Grasping

In this section, the proposed hierarchical reinforcement learning framework is applied to robotic grasping. Our goal is to learn multiple grasp policies π_l^k that allow to perform grasps across K different grasp types without prior knowledge of the reward function. The object to grasp serves as the global context of the scene, and potential grasp locations $\mathbf{s}_i^k$ are estimated for each grasp type based on point cloud information of the object. To facilitate the generation of potential grasp locations, a contact database $\mathscr{C}^k$ is maintained for each grasp type. Given the option-dependent contexts $\mathbf{s}_i^k$, the algorithm proceeds as before. An overview of the complete approach is given in Fig. 3.1.
Section 3.3.1 explains the procedure for generating grasp locations in more detail. Subsequently, Section 3.3.2 highlights some further extensions based on prior knowledge about grasping.

3.3.1 Grasp Candidate Generation

Grasp location candidates are generated by locally matching a given object point cloud to contact parts collected from successful grasps. Assuming a database of M contact parts $\mathscr{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_M\}$ is given, the ICP algorithm [55] can be used to find the best match between the point cloud of an object and contact part $\mathbf{C}_j$. The method was proposed before in this context by [7].
Once an object point cloud $\mathbf{P}$ is obtained, we randomly sample a partial point cloud $\mathbf{p}_i \subset \mathbf{P}$. ICP iteratively estimates a homogeneous transformation matrix $\mathbf{H}_{ij}$, such that the transformed point cloud $\mathbf{H}_{ij}\mathbf{C}_j$ is most similar to $\mathbf{p}_i$ according to matching error d_{ij}. We choose the transformation matrix $\mathbf{H}_{ij^*}$ with the lowest error across all stored contact point clouds $\mathbf{C}_j \in \mathscr{C}$. A potential grasp part $\hat{\mathbf{p}}_i \subset \mathbf{P}$ on object point cloud $\mathbf{P}$ is then found in the vicinity of $\mathbf{H}_{ij^*}\mathbf{C}_{j^*}$. See Fig. 3.2 for

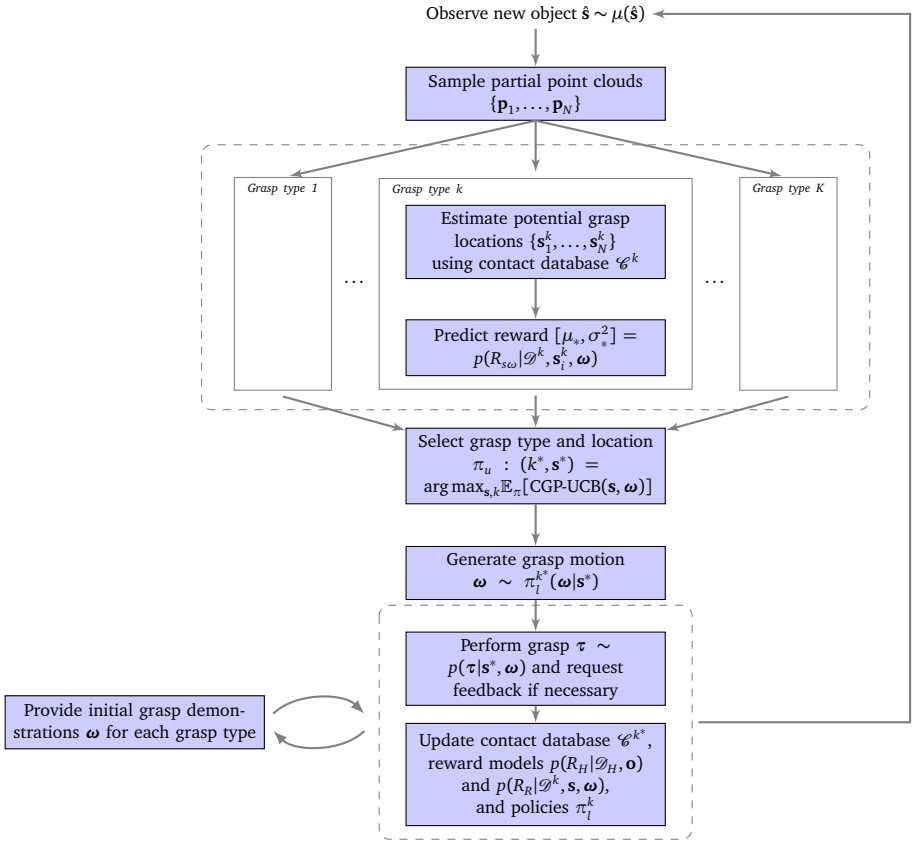


Figure 3.1: Overview of the approach applied to grasping. **Initialization:** Each grasp type k is initialized from human grasp demonstrations. The rollout data is stored to update the reward models $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$, $p(R_o|\mathscr{D}_\succ, \mathbf{o})$), and lower-level policies π_l^k. Finger contact information are extracted and stored in database $\mathscr{C}^k$. **Learning:** If a new object is presented, N partial point clouds are sampled from the whole point cloud of the object. These are used to estimate potential grasp locations $\{\mathbf{s}_i^k\}_{i=1,\ldots,N}$ for each grasp type k by matching the point clouds with stored contact templates from database $\mathscr{C}^k$. The grasp locations $\mathbf{s}_i^k$ and grasp types k are evaluated by context-parameter reward model $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$. Based on the estimated rewards, a grasp type k^* and location $\mathbf{s}^*$ are selected according to upper-level policy π_u. Next, a grasping motion is generated and executed given motion parameters $\boldsymbol{\omega} \sim \pi_l^{k^*}(\boldsymbol{\omega}|\mathbf{s}^*)$. Feedback about the outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$ is requested if the information gain with respect to the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ is large. Finally, the grasp types are updated accordingly.

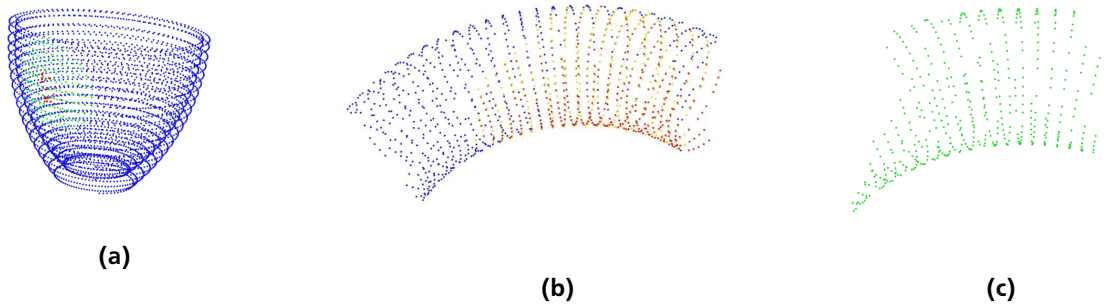
results of the grasp part estimation using ICP. In order to reduce the dimensionality, we represent each grasp part $\hat{\mathbf{p}}_i$ by a grasp location vector $\mathbf{s}_i$ consisting of the center and normal vectors extracted from $\hat{\mathbf{p}}_i$ for every finger. By repeating the procedure N times, a set of potential grasp locations is produced. The procedure is summarized in Algorithm 2. It is applied to each grasp type separately, where each grasp type maintains its own set of contact parts.
Since grasp locations are estimated locally, this approach is also feasible on a real system where a complete point cloud model might not be available.

3.3.2 Incorporation of Domain Knowledge

It is possible to make further changes to the general framework by exploiting prior knowledge about grasping. In this section, two such adjustments are presented. First, we utilize the matching error d_{ICP} of each grasp location candidate $\mathbf{s}_i^k$ from the ICP algorithm to guide the selection of a grasp type and location. The matching error is included directly into the acquisition function of the upper-level policy π_u, yielding

$$\pi_u : (\mathbf{s}^*, k^*) = \arg\max_{\mathbf{s},k} \mathbb{E}\Big[\mu_R(\mathbf{s}, \boldsymbol{\omega}) + \sqrt{\beta}\sigma_R(\mathbf{s}, \boldsymbol{\omega})\Big] + \gamma d_{\text{ICP}}, \tag{3.17}$$

where $\gamma > 0$ is a scaling constant. Second, we filter out grasps that are either infeasible given the robot kinematics or lead to undesirable object collisions when the trajectory is simulated. Since the expectation in Eq. 3.17 is approximated by many samples from the lower-level grasp policy π_l^k, it is computationally very expensive to exclude grasps directly



(a) (b) (c)

Figure 3.2: Grasp part estimation. **(a)** Contact points (red) and contact part in the neighborhood of contact points $\mathbf{C}$ (green) on the point cloud of an object $\mathbf{P}$ (blue). If the contact led to a successful grasp, the contact part is added to dataset $\mathscr{C}$. **(b) - (c)** ICP result for partial point cloud $\mathbf{p}_i$ of an object (blue). The matched contact part $\mathbf{C}_\mathbf{j}$ from dataset $\mathscr{C}$ is shown in red, whereas the best transformation of the ICP algorithm $\mathbf{H}_{ij^*}\mathbf{C}_{\mathbf{j}^*}$ is highlighted in yellow. The estimated grasp part $\hat{\mathbf{p}}_i$ is shown in green.

Algorithm 2 Estimation of grasp location candidates [7]

Input: number of grasp candidates N, object point cloud $\mathbf{P}$.
Initialization: store contact point clouds $\mathscr{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_M\}$ from successful grasps
for $i = 1$ to N **do**
  Randomly sample partial point cloud $\mathbf{p}_i \subset \mathbf{P}$
  for $j = 1$ to M **do**
    Perform $[d_{ij}, \mathbf{H}_{ij}] = \text{ICP}(\mathbf{p}_i, \mathbf{C}_j)$ between target point cloud $\mathbf{p}_i$ and source point cloud $\mathbf{C}_j \in \mathscr{C}$
  end for
  Compute $j^* = \arg\min_j d_{ij}$
  Estimate potential grasp part $\hat{\mathbf{p}}_i \subset \mathbf{P}$ in the vicinity of $\mathbf{H}_{ij^*}\mathbf{C}_{j^*}$
  Compute center $\mathbf{x}_i$ and normal vector $\mathbf{n}_i$ of grasp part $\hat{\mathbf{p}}_i$ for each finger f: $\mathbf{s}_i = [\mathbf{x}_i^1, \mathbf{n}_i^1, \ldots, \mathbf{x}_i^F, \mathbf{n}_i^F]$
end for
return grasp candidates $\mathscr{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$

at the upper-level policy. Instead, we reject parameter samples $\boldsymbol{\omega} \sim \pi_l^{k^*}(\cdot|\mathbf{s}^*)$ after a grasp type k^* and location $\mathbf{s}^*$ have been selected when the generated parameters lead to undesired behavior. Once a feasible grasp configuration is found, the grasp is executed and the algorithm proceeds as before.

4 Experiments and Results

In this chapter, our approach is evaluated through experiments on two different tasks. In Section 4.1, we examine various properties on a simple ball throwing toy task. Afterwards, the performance of our approach is assessed on a simulated grasping task in Section 4.2. The chapter closes with a discussion of the results in Section 4.3.
In order to allow for detailed evaluations, we synthesize preference feedback as follows. Given two subsequent outcomes $\mathbf{u}$ and $\mathbf{v}$, preference $\mathbf{u} \succ \mathbf{v}$ is generated if $R(\mathbf{u}) + \epsilon_u > R(\mathbf{v}) + \epsilon_v$, and $\mathbf{v} \succ \mathbf{u}$ otherwise. $\epsilon \sim \mathcal{N}(\epsilon|0,1)$ is standard Gaussian noise and $R(\cdot)$ is a task-specific reward function.

4.1 Ball Throwing Task

In this section, we examine various properties of our approach through experiments on a ball throwing toy task. First, the task is explained in more detail in Section 4.1.1. The results of our experiments are presented afterwards in Section 4.1.2.

4.1.1 Task Description

The goal of this task is to learn how to throw a ball to a goal position $\mathbf{p}_{\text{goal}} \in \mathscr{R}^2$ on a hilly landscape given the horizontal coordinate of the goal position as context $s \in [2, 10]$. The simulated robot consists of three revolute joints of equal length $l_1 = l_2 = l_3 = 0.2m$. Depending on the context, the robot can decide between $K = 2$ options. The robot can throw the ball either from left ($\mathbf{p}_{\text{base}} = [-4, 0]^T$) or right ($\mathbf{p}_{\text{base}} = [16, 0]^T$) of the landscape. An example of a ball throw is depicted in Fig. 4.1. This setting is a special case of our algorithm, where the context is the same across all options. Hence, the task of the upper-level policy π_u reduces to selecting the best option k given the current context s. Each of the two options uses a separate lower-level policy $\pi_l^k(\boldsymbol{\omega}|s)$. The motion parameters $\boldsymbol{\omega} \in \mathscr{R}^6$ consist of the joint angles and velocities of each joint when releasing the ball. After the ball has been released, it is assumed to be only affected by gravity. The trajectory of the ball is given by

$$\mathbf{p}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} p_{x_0} + v_0 \cos\beta\, t \\ p_{y_0} + v_0 \sin\beta\, t - \frac{g}{2} t^2 \end{bmatrix}, \tag{4.1}$$

where t is time, $g = 9.81 m/s^2$ is the gravitational acceleration, $\mathbf{p}_0 = [p_{x_0}, p_{y_0}]^T$ is the initial ball position, β is the initial throwing angle, and v_0 is the initial ball speed at ball release. Eliminating t allows to express the horizontal coordinate of the ball position in terms of the vertical coordinate, i.e.

$$p_y(p_x) = p_{y_0} + (p_x - p_{x_0}) \tan\beta - \frac{(p_x - p_{x_0})^2 g}{2 v_0^2 \cos^2\beta}. \tag{4.2}$$

To make use of Eq. 4.2, it is necessary to relate the joint parameters $\boldsymbol{\omega}$ to the release parameters $\{\mathbf{p}_0, \beta, v_0\}$ of the trajectory. Given the forward kinematics equations of the robot, we calculate the initial ball position $\mathbf{p}_0$ and velocity $\mathbf{v}_0$, assuming they coincide with the end-effector position and velocity at ball release. The velocity vector $\mathbf{v}_0 = [v_{x_0}, v_{y_0}]^T$ can then be decomposed as follows:

$$\beta = \tan^{-1} \frac{v_{y_0}}{v_{x_0}}, \tag{4.3}$$

$$v_0 = ||\mathbf{v}_0||_2. \tag{4.4}$$

The true reward R of a throw $\boldsymbol{\tau}$ with context s is given by

$$R(\boldsymbol{\tau}, s) = -||\mathbf{p}_{\text{goal}} - \mathbf{p}_{\text{hit}}||_2 - \frac{1}{2} m v_0^2 + c \tag{4.5}$$

where $\mathbf{p}_{\text{hit}}$ is the ball position after it has hit the ground, $m = 4kg$ is the mass of the ball, and $c > 0$ is a scaling constant. The first term measures the distance to the goal position, whereas the second term equals the work that is necessary to accelerate the ball to v_0. As shown in Fig. 4.2, in general higher rewards can be obtained by choosing the option that is closer to the context. Since the true reward function is not provided to the learner, we learn an outcome reward model $p(R_H|\mathscr{D}^k, \mathbf{o})$ from outcomes $\mathbf{o} = [\Delta\mathbf{x}, v_0]^T$, where $\Delta\mathbf{x} = ||\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{hit}}||_2$ is the distance to the goal position.

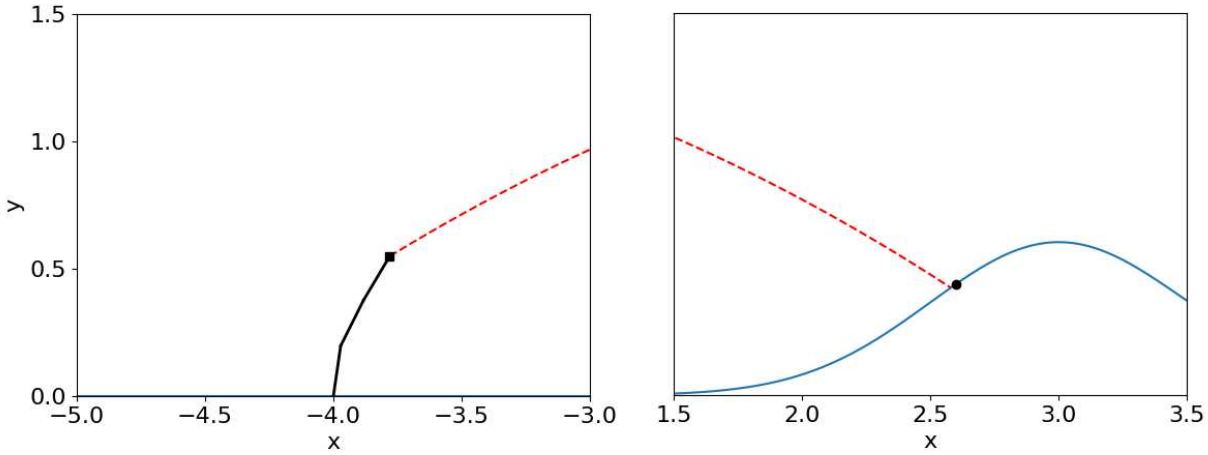


Figure 4.1: Ball throwing example. **Left**: The robot is mounted on the left side of the landscape at $\mathbf{p}_0 = [-4, 0]^T$ and throws the ball (red dotted line). **Right**: The ball lands close to the goal position $\mathbf{p}_{\text{goal}} = [2.6, 0.44]^T$ indicated by the black dot.

4.1.2 Experiments

In the first experiment, we performed four trials with 500 rollouts. The reward models and policies were initialized from 25 random rollouts, where contexts for the left policy were uniformly sampled from the left side of the landscape, $s \sim U(2, 6)$, and contexts for the right policy were sampled from the right side, $s \sim U(6, 10)$. Subsequently, the policies are updated after every 5 rollouts while the reward models are updated after every 50 rollouts.
The goal of the first experiment is to show that the complexity of the approach is justified. If we can achieve equal performance with less complexity, then we should opt for the simpler approach. We compare our algorithm to two baseline approaches:

- We only learn a single option, i.e. $K = 1$. This modification makes the use of a hierarchical architecture obsolete as we do not need to choose between policies. We use the left policy for all rollouts, and provide contexts uniformly sampled from the landscape for initialization, i.e. $s \sim U(2, 10)$.

- We remove the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ and directly learn the reward models $p(R_{s\omega}|\mathscr{D}^k, s, \boldsymbol{\omega})$ from human preferences.

The results are shown in Fig. 4.3. Our approach achieves the best performance with average rewards of around 1.9 after 175 rollouts, and a final average reward of 2.1 after 500 rollouts. It also shows a more stable performance compared to the baselines as indicated by the smaller error bars.
If no outcome reward model is used, the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, s, \boldsymbol{\omega})$ have to be learned directly from preferences, which is much harder due to the higher dimensionality of the input space and uncertainty about the outcomes. Because errors in the reward estimation subsequently misguide the policy search, the performance of this variant is worse and suffers from high variance.
If only a single policy is used, i.e. $K = 1$, the optimal performance is lower than the optimal performance with two policies by construction (see Fig. 4.2). Therefore, it is not surprising that our approach outperforms the modified version with only one policy. As depicted in Fig. 4.3b, our approach chooses the correct policy 97% of the time after 500 rollouts. This percentage drops to 75% on average if no outcome reward model is used. The results suggest that learning multiple policies and a separate outcome reward model are important for solving the task.
One of the reasons for including an outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ is that it can be learned quickly from outcomes $\mathbf{o}$. Once the outcome reward model has been learned, it can inform the context-parameter reward models $p(R_{s\omega}|\mathscr{D}^k, s, \boldsymbol{\omega})$ and lower-level policies π_l^k for each option k. Fig. 4.4 supports this reasoning. It compares the true reward distribution with respect to the outcome features to the estimated reward distribution as predicted by the outcome reward model after 100 rollouts. The estimated reward distribution is similar to the true one, although it differs in scale and curvature. Because the outcome reward model is learned from preferences, differences in scale can occur without loss of performance if the estimated latent rewards explain the preferences equally well. Similarly, the different curvature in regions of very small and very large velocities is not problematic per se. Keep in mind that we are not interested in estimating the whole reward function but only those areas where we actually observe data in practice. While most of the outcomes

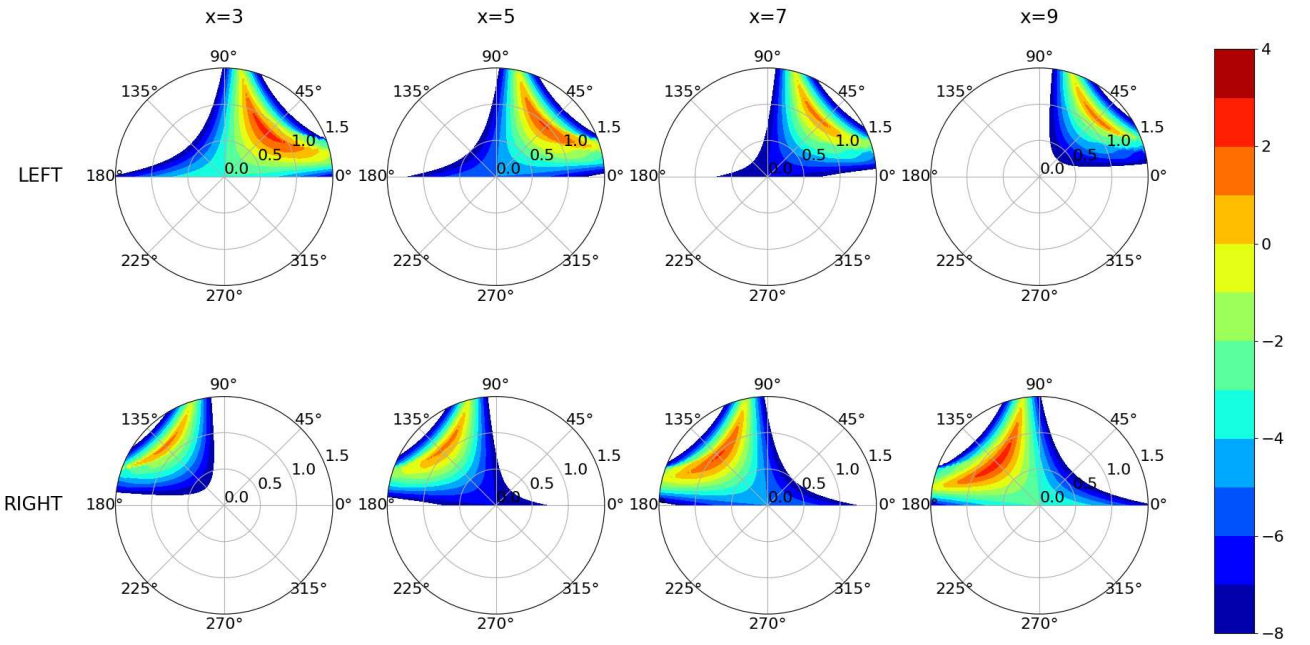


Figure 4.2: True rewards for left policy (top row) and right policy (bottom row) throws given different throwing angles and velocities in task space at ball release. Release positions are fixed at $\mathbf{x}_{0L} = [-4, 0]^T$ and $\mathbf{x}_{0R} = [16, 0]^T$. Rewards were measured at distinct contexts $s \in \{3, 5, 7, 9\}$. Depending on the context, either of the policies is preferable. Note that during the experiments the true reward values are assumed to be unknown. Instead, we are only given preference feedback from the human.

with velocities close to 0 are not physically feasible and therefore never observed, outcomes with high velocities are far from optimal and thus rarely seen.

In the next experiment, we evaluate the active learning component proposed in Section **??**. We compare the number of requested queries as well as task performance using $\lambda = 0.3$, $\lambda = 0.5$ and $\lambda = 0.9$ for 200 rollouts in four trials. As shown in Fig. 4.5a, the number of feedback requests is reduced significantly in all three cases. For $\lambda = 0.3$ the amount of queries is cut to about 60, whereas a threshold of $\lambda = 0.9$ leads to a reduction to about 10 requests on average. As the outcome reward model becomes more accurate, the number of feedback requests is reduced over time. As depicted in Fig. 4.5b, all variants chose the correct policy more than 95% of the time after 100 rollouts. Surprisingly, the algorithm is most confident for $\lambda = 0.9$ (few preference requests), whereas for $\lambda = 0.3$ it fails to choose the correct policy in 10% of the cases again after 150 rollouts.

The task performance for the different λ values is shown in Fig. 4.6. In all three cases, the performance of the system surpasses mean rewards of about 1.5 after 100 rollouts. For $\lambda = 0.9$ (few preference requests), the variance of the obtained rewards shrinks most quickly. However, the performance later saturates at 1.85. If more preferences are requested ($\lambda \geq 0.5$), the final performance after 200 rollouts reaches 2.05 on average. At first glance, these findings contradict our earlier observations from Fig. 4.5a, where the algorithm was best able to choose the correct policy for $\lambda = 0.9$. The result suggests that for $\lambda = 0.9$ the outcome reward model merely learns to evaluate outcomes in terms of the required energy, whereas it underfits with respect to goal distance. Such a reward model is sufficient to decide which lower-level policy to choose, but is not accurate enough to learn optimal lower-level policies.

4.2 Robotic Grasping Task

After the ball throwing toy task presented in the previous section, we now move on to a robotic grasping task. The experimental setup is described in Section 4.2.1, followed by the results in Section 4.2.2.

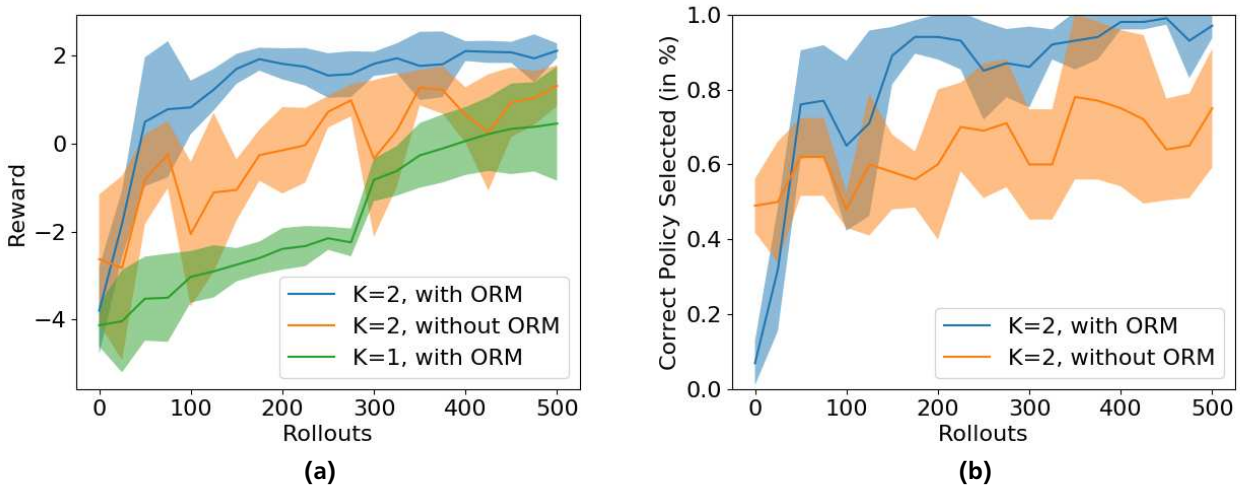


Figure 4.3: Performance on ball throwing task compared to baselines. Our approach (blue) uses $K = 2$ options and a separate outcome reward model (ORM). The first baseline (yellow) omits the outcome reward model $p(R_H|\mathscr{D}_H, \mathbf{o})$ and directly learns the reward models $p(R_R|\mathscr{D}^k, s, \boldsymbol{\omega})$ from preferences. The second baseline (green) learns only a single lower-level policy that is always used. Each evaluation consists of 25 test throws averaged over four trials. Error bars denote standard deviation. **(a)**: Average reward. Our proposed algorithm outperforms the baseline approaches and quickly converges to an average reward of about 1.9. **(b)**: Average percentage of correct policy selection, where the correct policy is the policy that can achieve higher rewards (see Fig. 4.2). Our approach (blue) is able to select the correct policy more than 80% of the time after 150 rollouts, and almost always after 400 rollouts. In contrast to that, the learning progress is much slower when no outcome reward model is used (yellow).

4.2.1 Task Description

We apply our framework to learn pinch grasps, power grasps and medium wraps using the KUKA Light Weight Robot with a DLR/HIT II Hand in the V-REP simulator. The design of both the arm and the hand of the robot is based on the kinematics of the human. The arm has seven degrees of freedom, corresponding to three shoulder joints, an elbow joint, and three wrist joints. Its five-fingered hand consists of three joints per finger, resulting in 15 degrees of freedom for the end-effector. However, we do not directly learn the joint torques but we predict the goal position and orientation in task space and then interpolate between the initial configuration and the goal configuration of the robot to generate a trajectory. The joint torques can then be inferred through inverse kinematics of the robot.
The motion parameters $\boldsymbol{\omega} = [\mathbf{x}_{\text{pre}}, \mathbf{x}_{\text{grasp}}, \mathbf{q}_{\text{grasp}}]^T, \boldsymbol{\omega} \in \mathscr{R}^{10}$ consist of a pre-grasp and grasp position of the hand palm in task space, and a quaternion specifying the orientation during the grasp. The finger motion is fixed for every grasp type to keep the number of parameters small. Contexts $\mathbf{s} \in \mathscr{R}^{12}$ are restricted to the contact position and normal vector for thumb and index finger only. To generate preferences from simulated grasps, we assume the following true reward function R for grasp $\boldsymbol{\tau}$ given object $\tilde{\mathbf{s}}$:

$$R(\boldsymbol{\tau}, \tilde{\mathbf{s}}) = c_1 \delta_{\text{lift}} + \frac{1}{2} \log(\gamma_{\text{grasp}} \gamma_{\text{lift}}) - c_2 ||\mathbf{q}_{\text{grasp}} - \mathbf{q}_{\text{lift}}||_2, \tag{4.6}$$

where γ_{grasp} and γ_{lift} are the number of contacts when the fingers are closed and the hand is subsequently lifted (with $\log(0) := 0$), $\mathbf{q}_{\text{lift}}$ is the orientation of the object after it was lifted, and c_1 and c_2 are positive constants. $\delta_{\text{lift}} \in \{0, 1\}$ indicates whether the object was lifted at all. It is set to 1 if there are more than five contact points after lifting the object, $\gamma_{\text{lift}} > 5$, and 0 otherwise. The preferences are used to learn the outcome reward model $p(R_o|\mathscr{D}_\succ, \mathbf{o})$. For the outcome features $\mathbf{o} = \phi(\boldsymbol{\tau}) \in \mathscr{R}^3$, we use the horizontal displacement, the vertical displacement and the change in orientation of the object after the grasp.

4.2.2 Experiments

Throughout the experiments, each of the grasp types is initialized with 12 demonstrations, which are provided by fixing the motion parameters $\boldsymbol{\omega}$ by hand. For every subsequent rollout, an object point cloud is generated from which potential

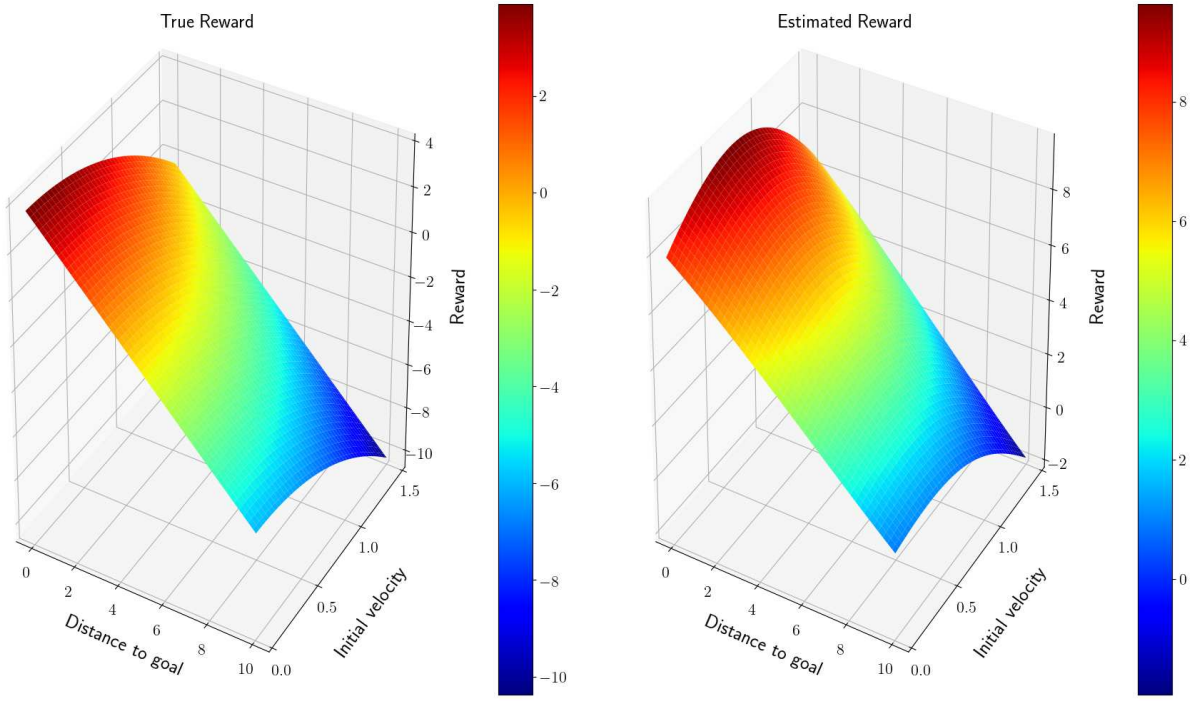
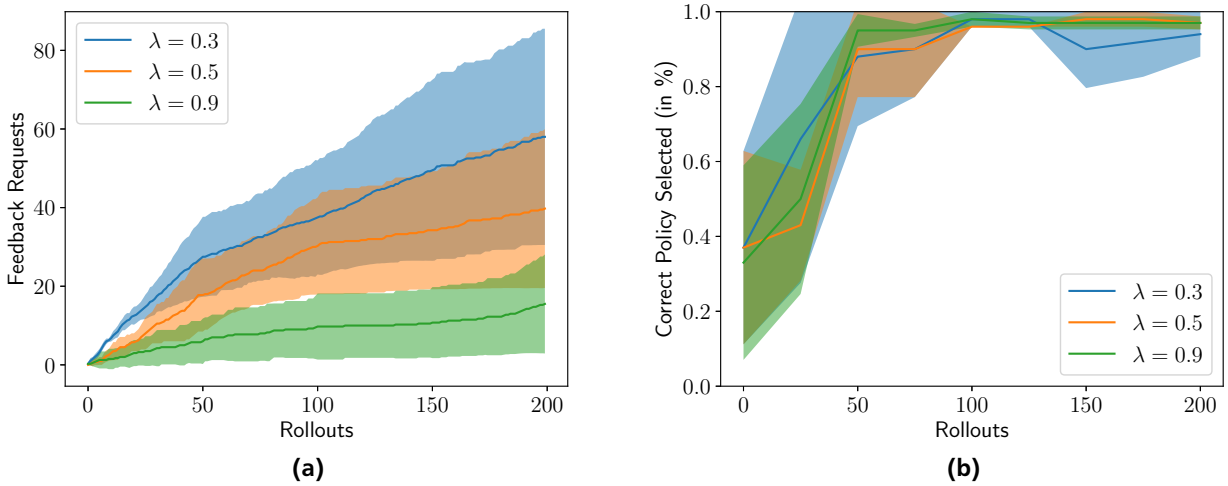


Figure 4.4: True reward distribution (left) compared to estimated reward distribution (right) given by outcome reward model $p(R_o|\mathscr{D}_{\succ}, \mathbf{o})$ after 100 rollouts. Outcome features $\mathbf{o} = \phi(\tau)$ of trajectory τ are defined as the distance to the goal $||\mathbf{x}_{goal} - \mathbf{x}_{hit}||$ and the initial velocity v_0. The estimated reward distribution is similar to the true one up to curvature and scale.



(a) **(b)**

Figure 4.5: Evaluation of active learning component for different λ values on the ball throwing task. **(a)**: Average number of feedback requests. In all cases, the amount of queries was reduced significantly. The higher λ, the less feedback was obtained. **(b)**: Average percentage the correct policy was selected. After 50 rollouts, the correct policy was chosen more than 90% of the time across all thresholds.

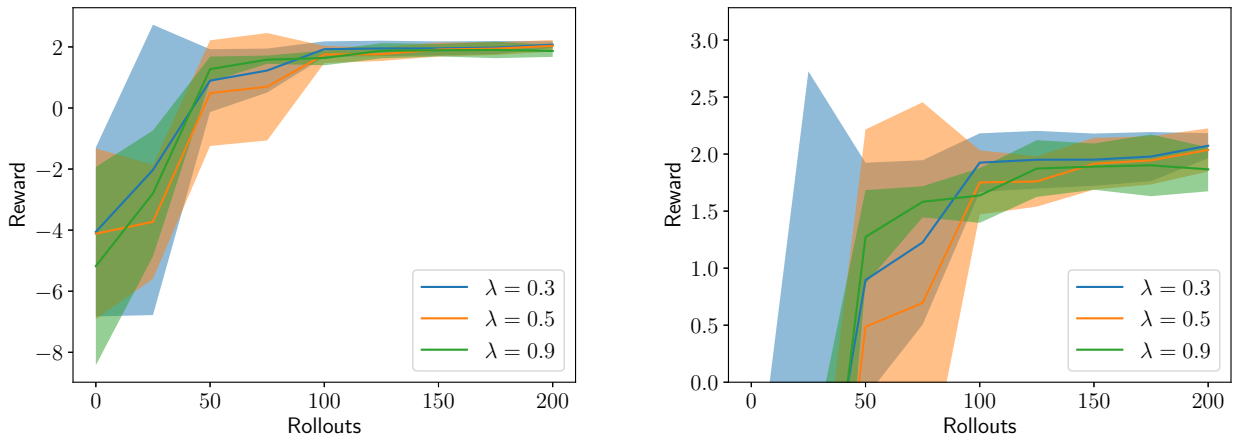


Figure 4.6: Performance on ball throwing task for different active learning thresholds λ. **Left**: In all three cases, average rewards of 1.5 and above are reached after 100 rollouts. **Right** (magnified version): The variance in the rewards decreases most quickly for $\lambda = 0.9$, but the algorithm converges prematurely with a mean reward of 1.85. In contrast, mean reward values of about 2.05 are achieved with lower λ values.

grasp locations are extracted per grasp type. Based on the expected rewards, the robot selects a grasp type k and location $\mathbf{s}$, and generates the motion parameters $\boldsymbol{\omega}$ for grasping the object. Once the fingers are closed, we lift the arm and ask for feedback if necessary. Each lower-level policy π_l^k is updated after every 12 rollouts with that policy. The reward models $p(R_o|\mathscr{D}_\succ, \mathbf{o})$ and $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ are updated after every three episodes, i.e. after every 36 rollouts.

In the first experiment, the goal is to learn how to grasp six different objects. The objects are shown in Fig. 4.7. Their point clouds are produced by converting the edges of the object models into a point cloud representation. Each of the three grasp types is initialized with demonstrations of two of the objects. The results for 250 rollouts averaged over three trials are depicted in Fig. 4.8a. As can be seen, the performance of the system improves from a success rate of 58% initially to 89% after 250 rollouts. Fig. 4.8b shows how often the correct policy was selected given the object, where the correct policy is defined as the one which was initialized with the presented object. However, keep in mind that some objects can be grasped with multiple grasp types. Directly after the initialization, our algorithm selects the correct policy for a given object 97% of the time. This value drops to 70% in the course of the learning process and recovers to 94% after 250 rollouts.

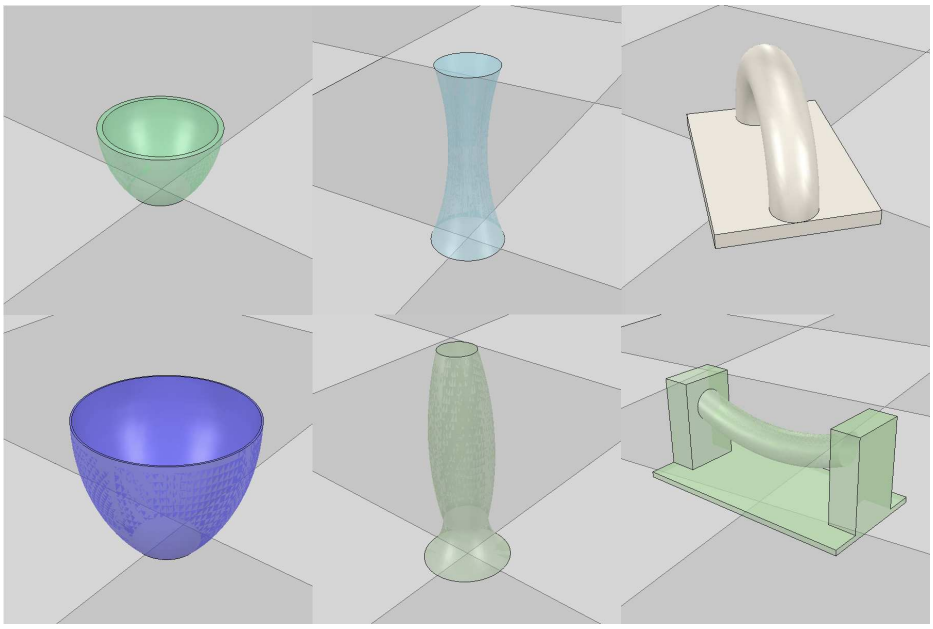
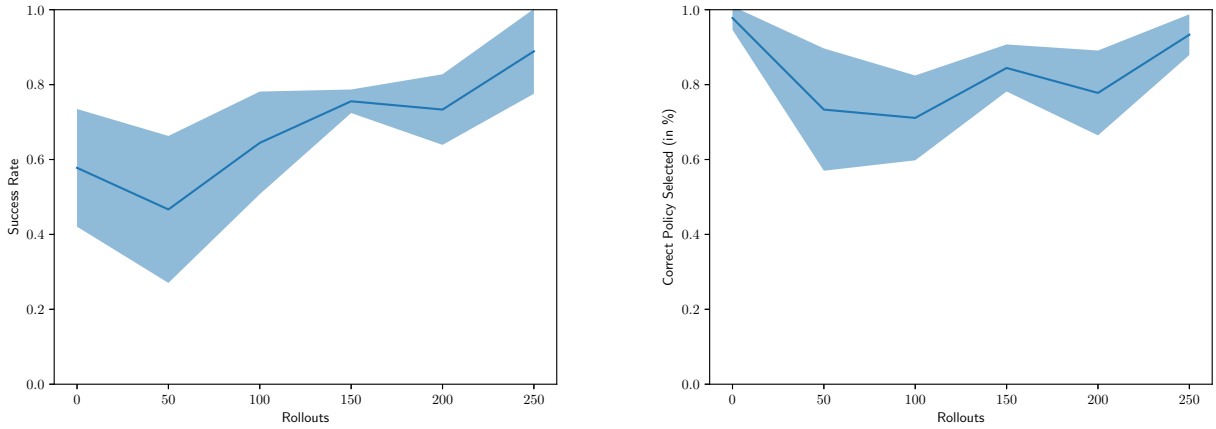


Figure 4.7: Known objects used for learning pinch grasps (left), power grasps (middle) and medium wrap grasps (right).



(a) (b)

Figure 4.8: Performance on a grasping task with known objects. **(a)**: Average success rate. The performance improves from 58% in the beginning to 89% after 250 rollouts. **(b)**: Average percentage where the correct policy was selected for grasping an object. Due to the strong prior from the ICP estimation, the upper-level policy almost always selected the lower-level policy correctly in the beginning. The percentage drops to 70% during training and finally recovers to 94%. Note that some of the objects can be grasped with more than one grasp type.

This behavior can be explained by the strong prior knowledge supplied in the beginning, followed by an exploration phase to learn and improve the models. The lower-level policies are well initialized from successful human demonstrations. In contrast, it is difficult to learn the outcome reward model $p(R_o|\mathscr{D}_{\succ}, \mathbf{o})$ from preferences since the initial rollouts are of similar quality. The predictions of the reward models $p(R_o|\mathscr{D}_{\succ}, \mathbf{o})$ and $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ therefore stay close to the GP prior. With predicted rewards $R_{s\omega}$ close to zero, the upper-level policy π_u relies heavily on the ICP matching error d_{ICP} (see Eq. 3.17) to select a context-option pair. The matching error is a good proxy for the reward of a grasp for known objects, but it is insufficient for grasping arbitrary objects. Over time, exploration causes more variance in the outcomes, and the estimates of the reward models are improved. As the reward models become more precise, the performance of the lower-level policies π_l^k improves as well.
The next experiment tests the ability of our algorithm to generalize to familiar objects, where familiar objects are defined as objects that are similar in shape compared to the objects used for initialization. We use 20 different objects across the three grasp types, see Fig. 4.9, and assign objects used for initialization randomly. In the former experiment, object point clouds were extracted from the edges of the object models. A side effect of this approach is that certain flat object regions that are not suited for grasping are excluded. Furthermore, this procedure fails for objects that consist of only flat faces, such as books or boxes. We therefore change the point cloud generation procedure for this experiment by sampling point clouds uniformly from the object surface.
Our results are shown in Fig. 4.10. Learning starts with an initial performance of 50%, which is reduced to about 25% after 50 rollouts. The success rate increases up to 53% again after 500 rollouts. Our results suggest that the algorithm is not fully able to generalize to familiar objects within 500 rollouts. Although the performance might further increase with more training, learning speed appears to be too slow for data-efficient grasping.

In the following section, the implications of the results obtained above are discussed in more detail.

4.3 Discussion

In the previous two sections, we performed several experiments on a ball throwing toy task as well as on a grasping task. On the first task, we were able to show that the performance improves when an outcome reward model is used. Furthermore, the number of feedback requests was reduced significantly by means of active learning. For the grasping task, we showed that our algorithm can learn to grasp a small set of known objects across three grasp types. However, we failed to achieve similar performance when the number of objects was increased to 20. In the following, we revisit several design choices of our algorithm in the light of the obtained results.

- Generation of preferences: The way the preferences are generated can have a significant influence on the results. In our case, the preferences were synthesized from a noisy reward function that was designed to produce reasonable

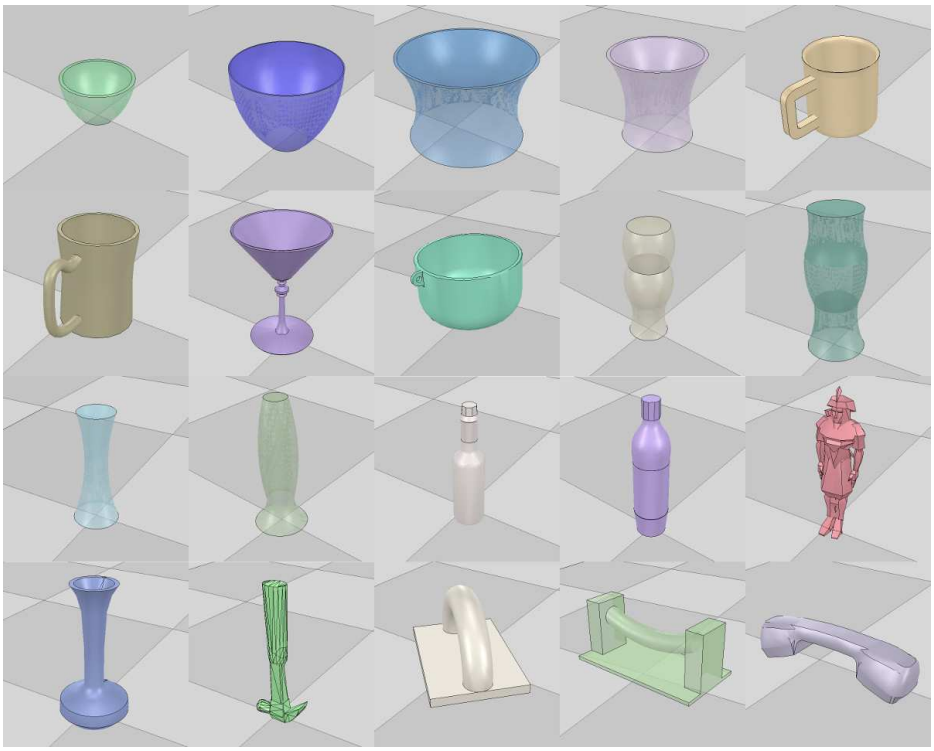


Figure 4.9: Familiar objects used for learning different grasping policies. Several objects models were taken from the Princeton Shape Benchmark [6]

preferences. The generation of preferences worked well for the ball throwing task because it is relatively easy to conceive a reward function for this setting. However, the original motivation of using preferences was to avoid exactly this kind of reward shaping. Indeed, it proved difficult to infer a good reward function in the grasping case.

- Acquisition of preferences: Pairwise preference feedback is requested only between subsequent rollouts, although theoretically many more comparisons are possible. In addition, other query schemes might be more informative. For example, one might ask for preference feedback between any two rollouts or between the previous best and the current rollout. The latter approach is a common strategy for obtaining preferences, but does not translate easily to our scenario as we are interested in learning multiple options for different contexts. Thus, there does not exist a single best rollout. Comparing two randomly selected instances would require to replay previous rollouts, which is indeed a viable possibility.

- Point cloud processing: The choice of method for generating object point clouds for the grasping task has an impact on the quality of the contexts that can be extracted and the type of objects that can be grasped. The most realistic scenario would be to extract all point clouds from a (simulated) depth camera, but this approach makes training more difficult. Osa et al. [7] generate the point clouds at training time by converting the edges of the object models (thus simplifying the problem) and employ a Kinect depth camera at test time on a real robot. This procedure can be seen as a form of curriculum learning, where the system first learns to solve a simpler problem in simulation and is then fine-tuned in a more difficult setting.

- Reward modeling: In all our experiments, we used the popular squared exponential kernel for constructing the covariance matrices of the reward models. However, this type of kernel is known to be too smooth for certain applications. For grasping in particular, one might assume that the true reward function is discontinuous as grasps that are successfully lifted are valued much higher than grasps that fail to lift the object. In fact, one could design a simple reward function that assigns a constant positive or negative reward to these outcomes, respectively, and several authors have employed such a reward function. But this binary reward scheme neglects differences between two successful grasps (or equivalently, two unsuccessful grasps), which can be captured by preference feedback. One might instead assume that the reward function $f(\mathbf{x})$ is composed of two additive processes $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, such that $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$. The first process $f_1(\mathbf{x})$ captures the reward assigned to whether the object was lifted and the second process $f_2(\mathbf{x})$ models variations in successful and unsuccessful grasps. In the GP framework,

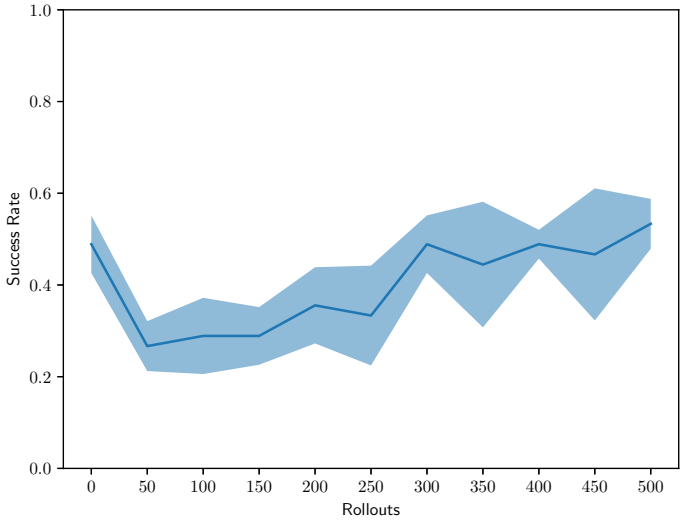


Figure 4.10: Performance on a grasping task with familiar objects. Initially, half of the tested objects were grasped successfully. The success rate decreased to 25% after 50 rollouts and reached a final performance of 53% after 500 rollouts.

modeling such a process is possible by adding the kernels of the two processes, i.e. $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$, where both kernels have different lengthscales [4].

Another design choice we made is to learn a separate context-parameter reward model $p(R_{s\omega}|\mathscr{D}^k, \mathbf{s}, \boldsymbol{\omega})$ for each option k. While this approach allows to deal with the different distributions of the datasets $\mathscr{D}^k$ by treating them independently, it disregards correlations between them. One way to incorporate dependencies across options is to employ multi-task learning approaches. In the context of GPs, several solutions have been proposed to address such a setting, and these might be worth investigating in the future.

5 Conclusion and Outlook

Grasping is a key manipulation skill for robots in order to be able to operate autonomously. Unfortunately, grasping is a notoriously difficult task to learn because of the intricacies of the real world. In particular, it is challenging to account for the large variety of objects that are present in our environment. One way to deal with this problem is to employ reinforcement learning, which allows to learn a grasping policy by trial and error. Therefore, information about previously unseen objects can be acquired over time in order to improve the learned grasping policy in an online fashion. In Chapter 3 of this thesis, such a reinforcement learning approach was presented. We proposed a general hierarchical reinforcement learning framework that allows to learn multiple lower-level policies across different options. At the upper level, a selection policy chooses the best context-option pair for a certain rollout while the lower-level policy of the selected option generates the action. Using this framework for grasping allows to learn versatile grasping motions for various grasp types, which are necessary to grasp a large variety of objects.
Due to a lack of good reward functions for grasping, we suggested to learn two kinds of reward models in Section 3.1: a context-parameter reward model for each option and a separate outcome reward model learned from preferences. While the former is useful for determining which context-option pair to choose for a given rollout, the latter can be learned quickly to evaluate the outcome of a rollout. In order to reduce the number of feedback requests to learn the outcome reward model, an active learning criterion was proposed that measures the expected change in the predictive posterior. Both types of reward models were integrated with the upper-level policy and the option-specific lower-level policies in Section 3.2. The upper-level policy was defined based on an acquisition function from the BO literature while the lower-level policies were learned through contextual REPS. In Section 3.3, the general approach was tailored towards robotic grasping. In particular, we adopted the ICP-based context estimation algorithm as proposed by Osa et al. [7].
In Chapter 4, we examined the performance of the proposed framework on a ball throwing task and a robotic grasping task. We were able to show that including the outcome reward model facilitates the learning process and improves the overall performance. Furthermore, we were able to reduce the amount of feedback requests significantly through the proposed active learning component. On the robotic grasping task, we successfully learned to grasp few known objects using three different grasp types. However, as more objects were introduced into the task, the proposed approach failed to learn how to grasp these objects within a reasonable amount of time.
The obtained results suggest that more work is necessary to learn how to grasp a wide range of different objects. An interesting avenue for future research would be to further investigate the employed reward models. For example, as discussed in Section 4.3, one might examine different choices of kernels that are better suited for the grasping domain. It would also be worth exploring other options of approximating the posterior of the preference learning reward model along with different parameter estimation techniques. Moreover, we would like to investigate alternative ways to generate grasp location candidates. Apart from these design considerations, more experiments on a real system are required to verify and extend the results obtained in this thesis. In particular, it would be desirable to obtain real preference feedback from a human, thereby avoiding the need to artificially synthesize preferences.

The obtained results suggest that the presented framework is a viable alternative for learning multiple policies without prior knowledge of the reward function. Nevertheless, we are still far away from teaching robots how to grasp objects with human-level performance.

Bibliography

[1] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv:1603.02199*, 2016.

[2] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *International Conference on Robotics and Automation*, pp. 3406–3413, 2016.

[3] H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *International Conference on Intelligent Robots and Systems*, pp. 2043–2050, 2012.

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[5] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. arXiv:1012.2599, 2010.

[6] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton shape benchmark," in *Shape Modeling International*, pp. 167–178, 2004.

[7] T. Osa, J. Peters, and G. Neumann, "Experiments with hierarchical reinforcement learning of multiple grasping policies," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2016.

[8] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis – a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[9] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, 2012.

[10] M. R. Cutkosky and R. D. Howe, *Human Grasp Choice and Robotic Grasp Analysis*, pp. 5–31. 1990.

[11] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[12] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 348–353, 2000.

[13] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, 2015.

[14] C. Ferrari and J. Canny, "Planning optimal grasps," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2290–2295, 1992.

[15] T. Feix, J. Romero, H. B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The grasp taxonomy of human grasp types," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 66–77, 2016.

[16] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

[17] E. Oztop, N. S. Bradley, and M. A. Arbib, "Infant grasp learning: a computational model," *Experimental Brain Research*, vol. 158, no. 4, pp. 480–503, 2004.

[18] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[19] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999.

[20] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[21] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670, 2000.

[22] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases.," *Science*, vol. 185, pp. 1124–31, 1974.

[23] D. C. Kingsley, "Preference uncertainty, preference refinement and paired comparison choice experiments," tech. rep., University of Colorado Boulder, 2006.

[24] L. L. Thurstone, "A law of comparative judgement," *Psychological Review*, vol. 34, pp. 278–286, 1927.

[25] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, "Template-based learning of grasp selection," in *International Conference on Robotics and Automation*, pp. 2379–2384, 2012.

[26] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning motion primitive goals for robust manipulation," in *International Conference on Intelligent Robots and Systems*, pp. 325–331, 2011.

[27] C. Daniel, O. Kroemer, M. Viering, J. Metz, and J. Peters, "Active reward learning with a novel acquisition function," *Autonomous Robots*, vol. 39, no. 3, pp. 389–405, 2015.

[28] R. Akrour, M. Schoenauer, and M. Sebag, *Preference-Based Policy Learning*, pp. 12–27. 2011.

[29] R. Akrour, M. Schoenauer, and M. Sebag, *APRIL: Active Preference Learning-Based Reinforcement Learning*, pp. 116–131. 2012.

[30] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *arXiv preprint arXiv:1706.03741*, 2017.

[31] A. Kupcsik, D. Hsu, and W. S. Lee, "Learning dynamic robot-to-human object handover from human feedback," in *Proceedings of the International Symposium on Robotics Research*, 2015.

[32] C. Wirth and J. Fürnkranz, "Preference-based reinforcement learning: A preliminary survey," in *Proceedings of the ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*, 2013.

[33] A. Morales, E. Chinellato, A. H. Fagg, and A. P. del Pobil, "An active learning approach for assessing robot grasp reliability," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 485–490, 2004.

[34] L. Montesano and M. Lopes, "Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 452–462, 2012.

[35] J. Fürnkranz and E. Hüllermeier, *Preference Learning*. New York, NY, USA: Springer, 2010.

[36] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 137–144, 2005.

[37] F. Mosteller, "Remarks on the method of paired comparisons: I. the least squares solution assuming equal standard deviations and equal correlations," *Psychometrika*, vol. 16, no. 1, pp. 3–9, 1951.

[38] D. J. C. MacKay, *Bayesian Methods for Backpropagation Networks*, pp. 211–254. 1996.

[39] H. J. Kushner, "A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, p. 97.

[40] V. T. J. Močkus and A. Žilinskas, "The application of bayesian methods for seeking the extremum," in *Towards global optimisation 2*, pp. 117–128, 1978.

[41] D. Lizotte, *Practical Bayesian Optimization*. PhD thesis, 2008. Ph.D. thesis.

[42] N. Srinivas, A. Krause, S. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015–1022, 2010.

[43] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.

[44] B. Settles, "Active learning literature survey," Computer Science Technical Report 1648, University of Wisconsin–Madison, 2009.

[45] J. Kober and J. Peters, *Reinforcement Learning in Robotics: A Survey*, pp. 579–610. 2012.

[46] C. Daniel, G. Neumann, and J. Peters, "Hierarchical relative entropy policy search," in *Fifteenth International Conference on Artificial Intelligence and Statistics*, vol. 22, pp. 273–281, 2012.

[47] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robots," in *The International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[48] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1, pp. 171–203, 2011.

[49] G. Neumann, "Variational inference for policy search in changing situations," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 817–824, 2011.

[50] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pp. 1401–1407, 2013.

[51] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, pp. 1607–1612, 2010.

[52] W. Chu and Z. Ghahramani, "Extensions of gaussian processes for ranking: semi-supervised and active learning," in *Neural Information Processing Systems Workshop on Learning to Rank*, 2005.

[53] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 2447–2455, 2011.

[54] A. Abdolmaleki, N. Lau, L. P. Reis, and G. Neumann, "Regularized covariance estimation for weighted maximum likelihood policy search methods," in *Humanoids*, pp. 154–159, IEEE, 2015.

[55] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.