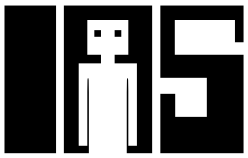
Machine Learning through Exploration for Perception-Driven Robotics

Machinelles Lernen zur Exploration in der Perzeptions-basierte Robotik
Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Herke van Hoof M.Sc. aus Zwolle, die Niederlande
Tag der Einreichung: 20. September 2016, Tag der Prüfung: 1. November 2016
Darmstadt, 2016 — D 17

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Marc Toussaint

TECHNISCHE
UNIVERSITÄT
DARMSTADT



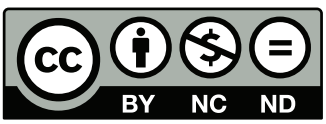
Machine Learning through Exploration for Perception-Driven Robotics
Machinelles Lernen zur Exploration in der Perzeptions-basierte Robotik

Genehmigte Dissertation von Herke van Hoof M.Sc. aus Zwolle, die Niederlande

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Marc Toussaint

Tag der Einreichung: 20. September 2016
Tag der Prüfung: 1. November 2016

Darmstadt, 2016 — D 17



Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation 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 3. November 2016

_____

(Herke van Hoof)

Abstract

The ability of robots to perform tasks in human environments has largely been limited to rather simple and specific tasks, such as lawn mowing and vacuum cleaning. As such, current robots are far away from the robot butlers, assistants, and housekeepers that are depicted in science fiction movies. Part of this gap can be explained by the fact that human environments are hugely varied, complex and unstructured. For example, the homes that a domestic robot might end up in are hugely varied. Since every home has a different layout with different objects and furniture, it is impossible for a human designer to anticipate all challenges a robot might face, and equip the robot a priori with all the necessary perceptual and manipulation skills.

Instead, robots could be programmed in a way that allows them to adapt to any environment that they are in. In that case, the robot designer would not need to precisely anticipate such environments. The ability to adapt can be provided by robot learning techniques, which can be applied to learn skills for perception and manipulation. Many of the current robot learning techniques, however, rely on human supervisors to provide annotations or demonstrations, and to fine-tuning the methods parameters and heuristics. As such, it can require a significant amount of human time investment to make a robot perform a task in a novel environment, even if statistical learning techniques are used.

In this thesis, I focus on another way of obtaining the data a robot needs to learn about the environment and how to successfully perform skills in it. By *exploring* the environment using its own sensors and actuators, rather than passively waiting for annotations or demonstrations, a robot can obtain this data by itself. I investigate multiple approaches that allow a robot to explore its environment autonomously, while trying to minimize the design effort required to deploy such algorithms in different situations.

First, I consider an unsupervised robot with minimal prior knowledge about its environment. It can only learn through observed sensory feedback obtained though interactive exploration of its environment. In a bottom-up, probabilistic approach, the robot tries to segment the objects in its environment through clustering with minimal prior knowledge. This clustering is based on static visual scene features and observed movement. Information theoretic principles are used to autonomously select actions that maximize the expected information gain, and thus learning speed. Our evaluations on a real robot system equipped with an on-board camera show that the proposed method handles noisy inputs better than previous methods, and that

action selection according to the information gain criterion does increase the learning speed.

Often, however, the goal of a robot is not just to learn the structure of the environment, but to learn how to perform a task encoded by a reward signal. In addition to the weak feedback provided by reward signals, the robot has access to rich sensory data, that, even for simple tasks, is often non-linear and high-dimensional. Sensory data can be leveraged to learn a system model, but in high-dimensional sensory spaces this step often requires manually designing features. I propose a robot reinforcement learning algorithm with learned non-parametric models, value functions, and policies that can deal with high-dimensional state representations. As such, the proposed algorithm is well-suited to deal with high-dimensional signals such as camera images. To avoid that the robot converges prematurely to a sub-optimal solution, the information loss of policy updates is limited. This constraint makes sure the robot keeps exploring the effects of its behavior on the environment. The experiments show that the proposed non-parametric relative entropy policy search algorithm performs better than prior methods that either do not employ bounded updates, or that try to cover the state-space with general-purpose radial basis functions. Furthermore, the method is validated on a real-robot setup with high-dimensional camera image inputs.

One problem with typical exploration strategies is that the behavior is perturbed independently in each time step, for example through selecting a random action or random policy parameters. As such, the resulting exploration behavior might be incoherent. Incoherence causes inefficient random walk behavior, makes the system less robust, and causes wear and tear on the robot. A typical solution is to perturb the policy parameters directly, and use the same perturbation for an entire episode. However, this strategy tends to increase the number of episodes needed, since only a single perturbation can be evaluated per episode. I introduce a strategy that can make a more balanced trade-off between the advantages of these two approaches. The experiments show that intermediate trade-offs, rather than independent or episode-based exploration, is beneficial across different tasks and learning algorithms.

This thesis thus addresses how robots can learn autonomously by exploring the world through unsupervised learning and reinforcement learning. Throughout the thesis, new approaches and algorithms are introduced: a probabilistic interactive segmentation approach, the non-parametric relative entropy policy search algorithm, and a framework for generalized exploration. To allow the learning algorithms to be applied in different and unknown environments, the design effort and supervision required from human designers or users is minimized. These approaches and algorithms contribute towards the capability of robots to autonomously learn useful skills in human environments in a practical manner.

Zusammenfassung

Die Fähigkeit von Robotern Aufgaben in menschlichen Umgebungen zu erfüllen hat sich bisher weitgehend auf relativ einfache und spezifische Aufgaben, wie Rasenmähen und Staubsaugen, beschränkt. Als solche sind sie weit entfernt von den Robotern als Butlern, Assistenten, und Haushälter, die in Science-Fiction-Filmen dargestellt sind. Ein Teil dieser Differenz kann durch die Tatsache erklärt werden, dass die menschliche Umwelt enorm vielfältig und komplex ist. Diese Eigenschaften machen es sehr schwer für einen menschliche Entwickler alle Herausforderungen zu antizipieren, die einen Roboter konfrontieren können.

Roboter könnten stattdessen auf einer Weise programmiert werden, die es ihnen ermöglicht sich durch lernen an ihre Umgebung anzupassen. Dies würde die Notwendigkeit für den Entwickler des Roboters Umgebungen genau zu antizipieren unnötig machen. Viele der aktuellen Techniken für lernende Roboter, sind jedoch auf menschliche Annotation, Demonstrationen und Feinabstimmung angewiesen. Dadurch ist der Vorteil dieser Lerntechniken beschränkt.

In dieser Dissertation werde ich mich auf eine andere Art und Weise fokussieren die Daten zu erhalten, die ein Roboter zum lernen braucht. Durch die *Exploration* der Umgebung mittels der eigenen Sensoren und Aktoren, erhält ein Roboter Daten, die es ihm ermöglichen zu lernen erfolgreich in seiner Umgebung zu agieren. Ich werde mehrere Ansätze untersuchen, die es einem Roboter ermöglichen seine Umgebung autonom zu explorieren und gleichzeitig versuchen, den Entwicklungsaufwand für den Einsatz solcher Algorithmen in verschiedenen Situationen zu minimieren.

Erstes betrachte ich einen unüberwachten Roboter mit minimalem Vorwissen über seine Umgebung. Der Roboter beobachtet das sensorische Feedback, das durch die interaktive Exploration der Umgebung ausgelöst wird. Nur dieses Feedback ermöglicht ihm das Lernen. In einem Bottom-up, probabilistischen Ansatz versucht der Roboter die Objekte in seiner Umgebung durch Cluster-analyse mit minimalem Vorwissen auf Basis von visueller Merkmale und den beobachtete Bewegungen zu segmentieren. Informationstheoretische Prinzipien werden verwendet, um eigenständig Aktionen auszuwählen, die den erwarteten Informationsgewinn, und damit die Lerngeschwindigkeit, maximieren. Die Auswertungen auf einem realen Robotersystem mit On-Board-Kamera zeigen, dass das vorgeschlagenen Verfahren verrauschte Daten besser verarbeitet als bisherige Verfahren, und dass die Auswahl der Aktionen nach dem Informationsgewinnkriterium die Lerngeschwindigkeit tatsächlich erhöht.

Im Gegensatz zu diesem komplett unbeaufsichtigtem Setup, ist im Framework des verstärkenden Lernens eine schwache Form von Rückmeldung in Form von Belohnungssignalen vorhanden. Zusätzlich zu diesen schwachen Belohnungssignalen hat der Roboter Zugriff auf reichhaltige sensorische Daten, die selbst für einfache Aufgaben häufig nichtlinear und hochdimensional sind. Sensorische Daten können verwendet werden um ein Systemmodell zu lernen, aber in hochdimensionalen Datenräume erfordert dieser Schritt häufig manuell konstruierte Merkmale. Ich schlage einen Algorithmus zum verstärkenden Roboterlernen vor, der durch gelernte, nichtparametrische Modelle, Nutzenfunktionen und Strategien mit hochdimensionalen Darstellungen umgehen kann. Als solches ist der vorgeschlagene Algorithmus gut geeignet hochdimensionalen Signale wie Kamerabilder zu verarbeiten. Um zu vermeiden, dass der Roboter vorzeitig zu einer suboptimalen Lösung konvergiert, wird der Informationsverlust des Strategieoptimierungsschritts beschränkt. Diese Beschränkung stellt sicher, dass der Roboter die Auswirkungen seines Verhaltens auf die Umwelt weiterhin exploriert. Die Experimente zeigen, dass der vorgeschlagene 'non-parametric relative Entropy Policy Search' Algorithmus zu besseren Ergebnissen führt als vorherige Methoden, die entweder unbeschränkte Optimierungsschritte verwenden, oder versuchen den Zustandsraum mit universalen radialen Basisfunktionen abzudecken. Darüber hinaus ist das Verfahren auf einem Robotersetup mit hochdimensionalen Kamerabildern validiert.

Ein Problem bei typischen Explorationsverfahren ist, dass das Verhalten in jedem Zeitschritt unabhängig gestört wird, zum Beispeil durch die Auswahl von willkürlichen Aktionen oder Parametern. Als solches kann das entstehende Verhalten inkohärent sein, was zu ineffizienten Zufallsbewegungen, geringer Robustheit und Verschleiß am Roboter führt. Eine typische Lösung besteht darin, die gleiche Parameterstörung auf einer gesamten Episode zu verwenden, aber dies führt tendenziell zur einer Erhöhung der Anzahl benötigter Episoden. In dieser Dissertation wird eine Methode untersucht, die einen ausgewogeneren Kompromiss zwischen den Vorteilen beider Verfahren macht. Die Experimente zeigen, dass solche Kompromisse in verschiedenen Aufgaben und Lernalgorithmen von Vorteil sind.

Diese Arbeit fokussiert sich also auf Roboter die durch Exploration der Umwelt autonom lernen. Zu diesem Ziel werden neue Ansätze und Algorithmen für unbeaufsichtigtes und verstärkendes Lernen eingeführt: ein probabilistischer interaktiver Segmentierungsansatz, der 'non-parametric relative entropy policy search' Algorithmus und ein Framework für generalisierte Exploration. In diesen Ansätzen werden Entwicklungsaufwand und Überwachung durch Menschen minimiert, um sie einfach in unterschiedlichen Umgebungen anwenden zu können. Die Beiträge dieser Arbeit liefern einen Schritt in Richtung praktischer Lernverfahren für nützliche Roboterfähigkeiten in menschlichen Umgebungen.

Acknowledgement

I would like to thank my advisor Jan Peters for his advice and support throughout the writing of the thesis. He and my other co-authors, Bastian, Chris, Daniel, Filipe, Geri, Heni, Mike, Oli, Max, Nutan, Patrick, Simone, Tucker, Voot and Zhengkun also played a big role in performing the research and developing the insights that lead to this thesis through discussions and feedback.

Similarly, I would also like to thank my other office mates Daniel, Gregor, and Hany and all other colleagues for being part of a constructive and pleasant research environment with lots of discussions, suggestions, and constructive feedback. The staff at the Intelligent Autonomous Systems group provided essential support and helped pass bureaucratic hurdles. Thanks, Veronika and Sabine!

I'm grateful for Marc Toussaint, as expert on the topics in this thesis, to agree to be external committee member. Your input is greatly appreciated. I would like to thank the other members of my thesis committee, Professors Roth, von Stryk, Fürnkranz and Rothkopf, without whose time investment a thesis defense would not have been possible. Their feedback, as well as the criticism and suggestions from the anonymous reviewers of my publications, have helped improve the quality of this thesis.

Because life is not *just* work, I would not have been able to write this thesis without the support of all my friends, my colleagues, my family, and Jessica. Thank you!

Contents

List of Symbols

The following tables give a list of variables, constants, operators, functions, and functionals used throughout the thesis. Symbols that only pertain to a specific section are defined where they are used. Where possible, notation is kept consistent with prior work in the area. This unfortunately means that sometimes, the meaning of a symbol is overloaded. The correct meaning should in that case be obvious from the context.

Formatting

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\dot{x}$	Time derivative
x_i	Element i of $\mathbf{x}$
$x_{\setminus i}$	All elements *except* i of $\mathbf{x}$
$\hat{x}$	Estimate of x

Variables and constants

$\mathbf{s}$	State
$a, \mathbf{a}$	Scalar- or vector valued action.
$\mathbf{o}$	Observation
$\theta, \boldsymbol{\theta}$	Scalar- or vector valued parameter; an angle or vector of angles
γ	Discount factor, probability to *not* terminate an episode
$\mathcal{D}$	Dataset
ψ	An angle
$\boldsymbol{\Phi}, \boldsymbol{\Psi}$	Design matrix
$\mathbf{K}$	Gram matrix
τ	A torque
ω	Angular velocity
$\mathcal{S}$	State space
$\mathcal{O}$	Observation space
$\mathcal{A}$	Actions space
$\mathbb{R}$	Set of real numbers

Functionals

$\mathbb{E}_x[\cdot]$	Expectation with respect to x
$\mathbb{E}_x[\cdot\|y]$	Conditional expectation with respect to x
$\mathrm{D}_{\mathrm{KL}}(p\|\|q)$	Kullback-Leibler divergence from Q to P

Operators, Functions, and Distributions

$\text{Beta}(\cdot;\alpha,\beta)$	Beta distribution with shape parameters α and β
$\mathcal{N}(\cdot;\boldsymbol{\mu},\boldsymbol{\Sigma})$	Multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$
$\mathcal{U}(a,b)$	A uniform distribution with support $[a,b]$
$\text{H}(X), \text{H}(X\|y)$	Shannon- or differential entropy, conditional entropy
$\text{I}(X;Y)$	Mutual information between X and Y
$\|\cdot\|$	Norm; absolute value; cardinality of a set
$\|\|\cdot\|\|_p$	p-norm
$\langle\cdot,\cdot\rangle$	Inner product
$\mathbf{a}=\pi(\mathbf{s})$	Deterministic policy
$\pi(\mathbf{a}\|\mathbf{s})$	Stochastic policy
$\mu_\pi(\cdot)$	Steady-state distribution under policy π
$\mathcal{R}_{\mathbf{s}}^{\mathbf{a}}$	Reward function of state-action pair $(\mathbf{s},\mathbf{a})$
$\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}}$	Transition probability $p(\mathbf{s}'\|\mathbf{a},\mathbf{s})$
d	Differential operator
$\frac{\partial}{\partial x}$	Partial derivative with respect to x
$\boldsymbol{\phi}(\cdot),\boldsymbol{\psi}(\cdot)$	Feature function
$\text{k}(\mathbf{x},\mathbf{x}')$	kernel function
$\mathbf{k}(\mathbf{x})$	Vector of kernel evaluations between $\mathbf{x}$ and a given set of points
$\delta(\cdot)$	Some error; Dirac or Kronecker delta
$\cup$	Set union
$!$	Factorial
$\mathbb{1}$	Indicator function
$\otimes$	Kronecker product
$\mathbf{I}$	The identity matrix

Relations

$\stackrel{!}{=}$	Is defined to be
$\propto$	Is proportional to
$\sim$	Is distributed according to
$\approx$	Is approximately equal to
$\in$	Is member of
$\leftarrow$	Is assigned the value of
$\perp$	Is statistically independent of

1 Introduction

Robots are increasingly present in industries such as production and assembly. For example, manufacturing industries use 66 robots per 10,000 employees on average, with the automotive industry even having more than one installed robot for every ten employees [IFR, 2015]. More recently, robots have become more prominent in service industries, including domestic robots and care robots. Millions of robots have already been sold for domestics tasks or entertainment, and thousands of robots have been sold for handicap assistance and medical purposes [IFR, 2015].

However, domestic robots have, so far, been limited in their capabilities. Typically, they can perform only a single simple task, such as lawn mowing, vacuum cleaning, window cleaning, etcetera. Furthermore, this task can often only be performed in a specific environment. For example, lawns or floors should be easily traversable and free of larger objects for the robots to do their work. In these environments, very simple sensors and pre-programmed behaviors are sufficient to complete the robot's task.

As such, the current states of household robots is far away from the types of robot butlers, assistants, and housekeepers depicted in science fiction movies. These fictional robots are able to perform a variety of complex tasks in various non-stationary environments that require responding to complex sensor inputs. Performing such tasks in novel environments, robots will invariably encounter situations that no human designer could have thought of beforehand. Therefore, in such cases simple pre-programmed behaviors are not enough, and the robot will need additional information to succeed.

Robots could obtain this additional information in multiple ways. An engineer could program the robot for that particular situation, but that would be costly and time consuming, as such situations will keep coming up as robots are deployed in more and more environments. An alternative would be to allow the end user to specify the desired behavior or supply annotations. Robots could then learn from these datasets or demonstrations using machine learning techniques like imitation learning [Argall et al., 2009, Schaal, 1999]. However, the end user's time is valuable, such that, where possible, such interventions should be avoided.

Instead, this thesis will aim at developing methods that allow robots to learn autonomously with rich sensory input, without demonstrations or annotation labels. At the same time, such methods should minimize the time a human engineer needs to spend to program, tweak, and tune the robot's control software. Furthermore, they must be able to handle the noise, ambiguity, and unanticipated behavior that will occur in real world settings. The next sections will describe how these topics are addressed in this thesis. After that, we will give an overview of the contributions made in this thesis and a summary of the thesis's structure.

1.1 Learning through Autonomous Exploration

When a robot is faced with a problem it cannot yet solve, rather than relying on human programming, demonstrations, or annotations, robots could gather data to learn how to solve this problem themselves. Robots could, for example, try different actions and observe the actions' results to infer properties of the environment. Such interaction can reveal otherwise unobserved properties, and mimics the way human perception and learning relies on exploratory behavior [Bohg et al., 2016, Katz and Brock, 2008, Fitzpatrick and Metta, 2003]. Since such exploration does not, in general, require supervision by a human engineer or user, it is a critical ability for robots that are to be deployed in a large variety of challenging environments.

As such, studying the ability of a robot to gain understanding through exploration will be the main goal of this thesis. Multiple aspects of this ability are critical for success. For example, exploration might be largely self-driven when trying to infer properties of the environment, with the robot trying to expand its knowledge as much as possible. The amount learned can be quantified by using the differential entropy of the distribution over possible environments. When the environment is unambiguously identified, the entropy is minimal, while the entropy is maximal when the distribution over possible environments is uniform. The information gain (or Kullback-Leibler divergence) expresses how much the agent learns from a new observation [Little and Sommer, 2013, Mobin et al., 2014]. In my study on interactive segmentation, *I propose using the expected information gain as a criterion to select more informative actions*. This is in contrast to earlier work, that largely used heuristics or pre-determined actions. We show that choosing actions according to this criterion helps to learn about the environment faster.

In other situations, when the robot is trying to perform a task, exploration can be driven by task success. As such, exhaustively understanding the situation can be undesirable, as obtaining the minimal information needed to complete the tasks will result in faster task completion. Instead, exploration of the environment should be balanced against the exploitation of the aspects of the system dynamics that are already known. Since most systems are too complex to understand exhaustively with limited training time, exploring actions that are deemed likely to be optimal can be chosen to focus learning on promising regions of the state-action space. Learning new tasks can be formalized in the reinforcement learning frameworks, where common exploration heuristics take such an approach [Kaelbling et al., 1996, Deisenroth et al., 2013, Kober et al., 2013]. For example, in ϵ-greedy strategies a random action is chosen in a fraction ϵ of time step, with the action deemed optimal chosen in all other time steps. Boltzmann or soft-max policies choose an action according to the exponential of its expected usefulness, such that the action deemed optimal is chosen more often, but not exclusively. Stochastic controllers tend to choose actions according to a probability distribution centered on the action deemed optimal, yielding a similar effect. In systems with continuous actions, such as robots, only this last type of exploration is typically applicable.

It is often hard, however, to set the width of the policy distribution in such a way that learning converges to an optimal policy without prematurely stopping to explore.

In this thesis, *I extend a method that sets this trade-off in a principled manner using a bound on the information loss to the non-parametric case*. This yields the first algorithm that uses kernel methods to avoid manual feature design, while using an information-theoretic bound to control policy updates.

To yield meaningful information, multiple courses of actions need to be tried for enough time to evaluate their success. A disadvantage of stochastic controllers, is that exploratory actions are only tried for single time steps. Especially on robots, these high-frequency perturbations can lead to undesirable behavior such as high jerk, sensitivity to control delays, and 'washing out' of exploratory actions. Therefore, *I study a method that allows coherent exploration over multiple time steps*, generalizing existing notions of exploration.

1.2 Minimizing Design Effort through Principled Statistical Learning

Besides exploration, robot learning with minimal human input requires a number of other problems to be addressed. For example, methods that need heuristics, tuning, and tweaking tend to require high design effort every time a robot is deployed in a new situation, as the engineer coming up with heuristics and tuning parameters cannot conceive all possible future situations.

In this thesis, I will use multiple methods to reduce the design effort required for the robot to learn. Most importantly, throughout this thesis I will, where possible, *derive methods from first principles, such as universally valid principles from Bayesian statistics or information theory*. By using such principles, rather than situation-specific heuristics, it can be ensured that the method is valid regardless of the context in which it is applied. It also forces assumption made about the environment to be made explicit. Explicit assumptions are easier to verify than assumptions that are made implicitly by heuristics. Where there is no clear principle to, e.g., set parameters, I will attempt to give suggestions for data-dependent settings.

Furthermore, finding right representation for learning algorithms can take a lot of tuning and tweaking. Non-parametric methods can be used to avoid relying on a representation of a fixed size. For example, when the non-linear basis of a regression problem is not known, kernel methods can be used. Since kernel method results in a very rich representation, some kernels are applicable in a wide range of situations. Thus, choosing a kernel is generally easier than designing a feature representation. *I propose such an approach to learn control policies from high-dimensional sensor data in a statistically principled manner*. When trying to understand a cluttered environment, representing the environment often implies attaching properties to individual objects. However, the number of objects is not necessarily known. *Using a non-parametric technique, the number of objects in a segmentation task does not need to be specified in advance or using a designed heuristic*.

1.3 Learning from Ambiguous and Noisy Sensor Data

In any exploration scheme, it is important for robots to perceive the effects of their actions. To be able to function in arbitrary environments, robots cannot rely on an

instrumentation of the environment, such as markers or a tracking system. Instead, they obtain their information through on-board, noisy sensors. The observed effects of the robot's actions can be quite different from stimuli that might have been considered during the robot's design. Therefore, the robot cannot rely on a fixed perception module, but should directly process complex and high-dimensional sensory data from, e.g., its camera, range sensor, microphone, or touch sensors. This data is often noisy and ambiguous, and might arrive at the robot with a delay due to communication delays and other effects. Thus, even after sensing its environment, the robot might not be certain about the environment's state.

Simply assuming the state currently deemed most likely is the true state does not allow the robot to select actions that are robust to the uncertainty, and also does not allow the selection of actions that reduce the uncertainty. For these capabilities, it is important that the residual uncertainty after sensing the environment is represented internally. In this thesis, this will be done by employing a Bayesian approach. An initial prior distribution is updated by conditioning as observations are made, yielding a posterior distribution. For example, *I propose using a Bayesian non-parametric approach that allows us to represent a distribution over segmentations of a cluttered scene*, and *use a Gaussian process controller conditioned on executed actions, that retains uncertainty in areas far from any actions tried so far*. Communication delays are another source of uncertainty through partial observability. *We propose an algorithm for coherent exploration that is robust to such delays*.

1.4 Contributions

This thesis addresses learning through autonomous exploration using a robot's on-board sensors. This problem is at the intersection of machine learning, robotics, and (interactive) perception. In this section, I summarize the contributions made in each of these fields.

1.4.1 Algorithmic Contributions

Throughout this thesis, I propose new algorithms for machine learning through exploration. First of all, I propose an algorithm for quantifying the approximated information gain of actions in an interactive segmentation task. To my knowledge, this was the first interactive segmentation task to use such an information-theoretic selection criterion. Our experiments show, that actions selected in this manner yield faster learning performance. A Bayesian formulation allows us to marginalize important parameters, that consequently do not need to be set by hand.

Furthermore, I propose a reinforcement learning algorithm that combines a non-parametric approach with an information-theoretic bound on policy updates. To the best of my knowledge, this is the first algorithm to offer this combination of properties. We compared this algorithms to previous methods that employ a generic parametric representation or unbounded updates, and found that it performs better than those methods in various experiments.

The generalized exploration algorithm that I propose allows a unified view on step-based and episode-based exploration, and offers intermediate trade-offs. Although

purely step-based and purely episode-based methods have been studied in previous work, to my knowledge, this is the first algorithm that allows such a trade-off by allowing the parameters in successive time-steps to depend on each other. The value of such a trade-off is shown by the experiments, that show that a balanced trade-off often performs better in terms of average rewards than either extreme. Furthermore, coherent exploration policies apply lower jerk while exploring a larger part of the state-space than independent step-based exploration.

1.4.2 Robotics contributions

The methods that I develop in this thesis are inspired by problems in current robot learning methods, and are applied to scenarios with real and simulated robots. The first contribution to the field of robotics is to make robot learn better segmentations interactively. By interpreting the robot's observations in a principled manner, the Bayesian approach can handle occasional erroneous inputs better than previous methods. This scheme is, to the best of my knowledge, the first principled approach for doing so in interactive segmentation tasks. Selecting better exploratory actions furthermore helps keeping the amount of required interaction time, and with that robot wear and tear, to a minimum.

Additionally, I apply a non-parametric policy search methods to several robot learning tasks. I show that in this manner, a real-robot pendulum swing-up skill and a tactile stabilization skill can be learned from only sensory data (visual and tactile data, respectively). As such, the proposed method makes it easier to apply reinforcement learning methods on challenging robotics tasks with high dimensional input representations from robot sensors.

1.4.3 Robot perception contributions

Perception is a critical aspect of robotics. In novel environments, however, the robot does not have access to annotated example images that, for example, many computer vision methods need. Our interactive perception approach enables the robot to generate its own training data by interacting with the scene. Our research contributes by using a probabilistic approach to interactive perception, that allows a principled framework to integrate different noisy clues from observed movement and spatial distribution.

In reinforcement learning approaches, a robot's policy is usually based on a compact set of carefully designed features of its sensory input. I show that the non-parametric approach can learn robot controllers without an initial feature design step. Furthermore, features can be learned using a modified variational auto-encoder. The experiments show such learned features yield much better performance when high-dimensional sensory data is noisy.

Aspect	Chapter 2	Chapter 3	Chapter 4
Exploration	Maximize information gain for exploration	Limit information loss for exploration	Trade-off exploration coherence
Minimize design effort	Non-parametric inference of number of objects	Kernel-based learning to avoid specifying features and/or learn features	Facilitates exploration without heuristics
Real sensors	Probabilistic model to handles uncertainty	Learned transition model prevents fitting to noise	Coherent exploration more robust to communication delays

Table 1.1.: This tables summarizes the different aspects of this thesis's contributions per chapter.

1.5 Structure of the Thesis

This thesis will address aspects of each of the topics discussed in Sections 1.1-1.3: learning through autonomous exploration, minimizing design effort, and learning from ambiguous and noisy sensory data. Sensor-driven autonomous exploration will be the main focus of this thesis.

This thesis starts by considering how to efficiently learn about the composite structure of an unknown, novel environment. In Chapter 2, I propose using a Bayesian non-parametric sampling scheme to represent a posterior probability distribution over possible segmentation. It allows marginalizing important parameters, as well as the number of objects. Furthermore, the residual uncertainty can be used to select actions expected to yield the highest information gain. Our experiments show selecting actions in this manner allows segmentations to be inferred faster.

This information-gain strategy is well-suited to explore when no explicite manipulation goal is given. However, if the robot has an explicit manipulation goal, exploration should be traded off with exploiting knowledge so far to reach this goal. Learning policies for such problems is discussed in Chapter 3. This chapter introduces a policy search algorithm that is both non-parametric and uses information-theoretic constraints to limit policy updates. These properties make the algorithm suitable for learning policies from high-dimensional, redundant inputs directly, as I show in the experiments.

A possible disadvantage of policy search methods such as the introduced algorithms, is that exploration is often performed by perturbing actions at individual timesteps, yielding inefficient random walk behavior. To remedy this problem, I propose a unifying view on step- and episode based exploration in Chapter 4. This view introduces a trade-off parameter that can be set to yield familiar step- and episode based exploration algorithms, but it also enables a whole range of trade-offs that combine advantages from both step-based and episode-based paradigms.

The way different chapters of this thesis cover the aspects mentioned previously is summarized in Table 1.1. All chapters in this thesis can be read independently.

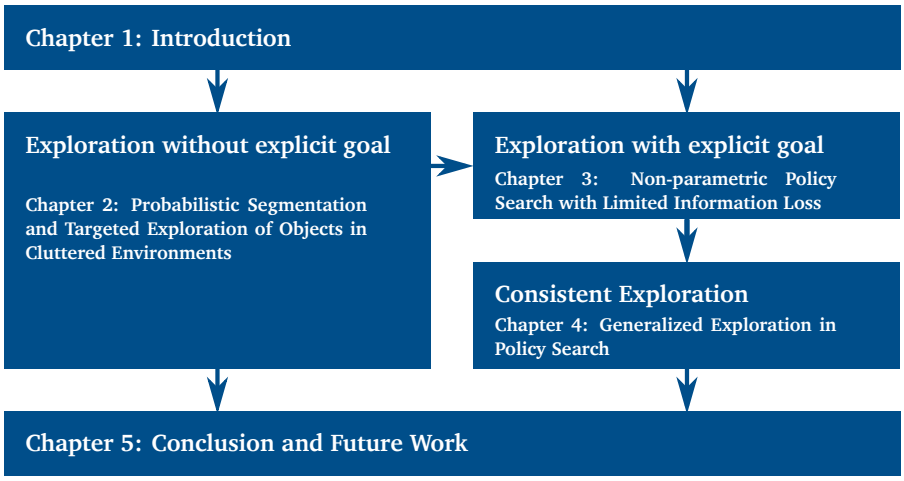


Figure 1.1.: Structure of this thesis. Recommended possible orders of reading the thesis's chapters are indicated by arrows.

However, Chapter 4 assumes basic knowledge of reinforcement learning terminology that is explained in Chapter 3, and extends the algorithm defined in that chapter. Therefore, it can be helpful to read Chapter 3 before Chapter 4. This structure of the thesis is illustrated in Figure 1.1.

2 Probabilistic Segmentation and Targeted Exploration of Objects in Cluttered Environments

In this chapter, we will address interactive segmentation. In interactive segmentation, a scene is segmented not just by using one or more passive camera images, but by using the potential of the robot to change the scene to resolve segmentation ambiguities. As such, it is an example of the wider field of interactive perception, where robots more generally use the capacity of interacting with the environment to make challenging perception problems easier [Bohg et al., 2016, Katz and Brock, 2008, Fitzpatrick and Metta, 2003].

Whereas most previous work on interactive segmentation has attempted finding the most likely segmentation, here, we propose representing a distribution over potential segmentation. This distribution represents the uncertainty of the robot, which could be exploited to, for example, select more robust action by avoiding highly uncertain areas. In our study, we will use the knowledge of the distribution for a different purpose: finding better explorative actions. That can be done by using a probabilistic model to calculate the expected information gain from observing the effect of all different actions to select the action for which this score is maximal.

The proposed action selection scheme is an improvement over earlier work, where robots performed manually selected actions, random actions, or actions chosen by a heuristic. In contrast, the method we propose is based on information-theoretic principles. Choosing actions with maximal information gain has furthermore been shown to minimize the Kullback-Leibler divergence between the true state of the world and the learned distribution [Little and Sommer, 2013, Mobin et al., 2014], a measure of the quality of the learned model.

2.1 Introduction

Many tasks require the recognition and manipulation of objects. Therefore, it is essential that robots assisting humans have such capabilities. Since human environments are unstructured, open-ended and dynamic, relying on a pre-specified database of possible objects is most likely insufficient. Rather, robots should learn about novel objects they encounter.

Supervised learning methods have been used to acquire object models [Pope and Lowe, 2000, Collet et al., 2009, Murase and Nayar, 1995, Mian et al., 2006]. However, such methods rely on a pre-structured data set provided by human teachers, limiting the applicability to newly encountered objects. Additionally, such objects often occur in clutter, whereas classical approaches require the object to be isolated, or

Figure 2.1.: We use a Mitsubishi PA-10 robot arm equipped with a force-torque sensor and an RGBD camera. The robot autonomously interacts with its environment to segment a scene into objects.

segmented by a (human) teacher [Mian et al., 2006, Murase and Nayar, 1995, Collet et al., 2009, Detry et al., 2009]. This requirement is usually not fulfilled when encountering objects in real, cluttered environments where such segmentations are not available.

Therefore, methods that can segment cluttered scenes are required as starting point for learning about the objects at hand [Li and Kleeman, 2008, Beale et al., 2011]. Such methods should be robust to noise and uncertainty (e.g., as induced by sensor- or manipulation failures and occlusions). Furthermore, minimizing the use of heuristics and hand-tuning makes the system more autonomous, by reducing the dependency on human time and effort.

In this chapter, we use a part-based approach to object segmentation. We use the robot's interaction with its environment to resolve segmentation ambiguities and to model objects robustly. Whereas most current work on robotic scene segmentation focuses on finding a single most likely segmentation, our approach retains a probability distribution over possible segmentations. We will evaluate this approach on real-world segmentation problems.

This probabilistic approach makes segmentation more robust to noise, and endows the robot with knowledge about the uncertainty in its model. Such knowledge can be exploited to choose the most informative action out of many possible actions using information-theoretic insights, whereas most recent work relied on prior training or human-crafted heuristics and domain knowledge. We will show that this criterion helps the robot to learn efficiently.

We extend this work by defining a new probabilistic model that allows motion clues to be integrated with static visual clues in a principled manner. Parameters of the likelihood model of the visual clues are learned from data rather than tuned by hand,

allowing the robot to transfer knowledge from previous scenes to the current task. We will show that using static visual clues in addition to motion information will enable our system to determine scene segmentations substantially faster.

2.2 Related Work

To learn the properties of novel objects in cluttered scenes, segmenting those images is usually a required first step [Li and Kleeman, 2008, Beale et al., 2011]. In this section, we will review various approaches to the scene segmentation problem and contrast our approach to this prior work. First, we will discuss methods that work with one or more static images. Subsequently, we will discuss several interactive approaches to solving this problem. Finally, we highlight a few important aspects of the problem and how these are addressed in prior work: dealing with noise and uncertainty, integrating visual with interactive clues, and efficient exploration.

2.2.1 Non-Interactive Visual Segmentation

Traditional segmentation approaches take a single image as input, using clues such as contrast, texture, or color [Carreira and Sminchisescu, 2012]. If 3D-data is available, 3D features such as surface normals or curvature might additionally be exploited [Strom et al., 2010, Erdogan et al., 2012]. However, visual or spatial boundaries need not always correspond to object boundaries [Hermans et al., 2012, Katz et al., 2010], so not all ambiguities can be resolved [Hermans et al., 2012, Kenney et al., 2009, Fitzpatrick and Metta, 2003, Bergström et al., 2011]. An alternative is to look at video streams [Pundlik and Birchfield, 2008]; however, in real-robot setups there may be too much (self-)occlusion for this strategy to be viable. Alternatively, a set of images containing the same objects [Herbst et al., 2011, Cho et al., 2008, Rother et al., 2006] is analyzed together. For example, co-segmentation and co-recognition methods [Cho et al., 2008, Rother et al., 2006] find image segments that occur in multiple images, usually with different backgrounds. Multi-scene analysis [Herbst et al., 2011], on the other hand, finds objects that moved between scenes with the same background.

In these approaches, the robot or system is a passive observer: it uses static images or waits for the environment to change. To learn autonomously, however, it is beneficial for a robot to cause its own changes in the environment.

2.2.2 Interactive Perception for Object Segmentation

Physical interaction with objects through pushing, grasping, or lifting enables a robot to learn about them. For learning how actions change the state of objects [Fitzpatrick and Metta, 2003, Modayil and Kuipers, 2008, Kraft et al., 2010, Sinapov et al., 2011, Katz et al., 2010, Griffith et al., 2009], interaction is even required, as the necessary information is not present in static (visual) data.

For learning the appearance and shape of individual objects [Fitzpatrick and Metta, 2003, Schiebener et al., 2013, Krainin et al., 2011a, Li and Kleeman, 2009, Ude

et al., 2008], interaction can also be helpful. Through interaction, a robot can obtain multiple views of a scene (for approaches like [Herbst et al., 2011]) autonomously. Besides, knowing what action was performed helps the robot to interpret ambiguous observations, such as movement of an object of interest when there is background movement as well [Beale et al., 2011].

To avoid the segmentation problem, objects can be physically separated from clutter by grasping [Ude et al., 2008, Kraft et al., 2010, Krainin et al., 2011b, Li and Kleeman, 2009, Natale et al., 2005]. Autonomously grasping novel objects, however, is non-trivial by itself and frequently requires knowledge of the object's geometry. Such knowledge is not present for novel objects.

An alternative to grasping is non-prehensile manipulation such as pushing. Employing non-prehensile manipulation for object segmentation was pioneered by Fitzpatrick and Metta [Fitzpatrick and Metta, 2003]. Their robot swept its arm across its workspace to detect objects using image differencing. Li and Kleeman [2008] refined this method using short, accurate pushes targeted at near-symmetrical objects. The accuracy of segmentations can be increased by accumulating information over time [Kenney et al., 2009].

These image differencing techniques estimate object membership per pixel. To estimate movement direction, a part based representation using an initial over-segmentation can be used [Beale et al., 2011]. In a subsequent stage, movement and object membership can be estimated for each of these parts. Alternative approaches use algorithms such as iterative closest point (ICP) to determine whether the tracked point cloud is a single rigid object [Chang et al., 2012, Hermans et al., 2012], or estimate the movement of trackable visual features [Schiebener et al., 2013, Chang et al., 2012, Sushkov and Sammut, 2011, Bergström et al., 2011, Katz et al., 2010, Hausman et al., 2013].

Interactive approaches offer powerful clues and enhanced robot autonomy. Therefore, we will employ an interactive segmentation method in our approach.

2.2.3 Dealing with noise and clutter

Data coming from robot sensors is frequently noisy and may be incomplete. The possibility of co-movement of multiple objects in cluttered scenes means that observed movement data may often be ambiguous. Thus, with a limited amount of experience, there is uncertainty about the true segmentation. Many of the discussed approaches do not handle occlusion and co-movement as they deal with only one object of interest, e.g., [Fitzpatrick and Metta, 2003, Li and Kleeman, 2008, Beale et al., 2011]. Noise is often ignored [Kenney et al., 2009, Fitzpatrick and Metta, 2003] or handled by requiring objects to move as rigid bodies during one or multiple actions [Bergström et al., 2011, Hermans et al., 2012, Cho et al., 2008, Hausman et al., 2013]. In these approaches, it is not clear how to deal with uncertainty, e.g., from occlusions or with pushes resulting in multiple adjacent objects moving as according to the same homogeneous transform.

The minimization of inconsistent movement is an alternative approach [Herbst et al., 2011]. This approach assumed objects are connected components, which does

not always hold in the presence of clutter and occlusions. Beale et al. [2011] employed a probabilistic method for correlating a segment's movement with that of the end-effector, but considered only a single object of interest.

Defining a probability distribution over the complete set of possible segmentations of all objects in the scene means that uncertainty can be quantified properly. We expect such an approach to result in robustness in the presence of failures and occlusion. Knowing the uncertainty also allows actions to be selected more robustly. For example, when grasping an object, parts whose object membership is uncertain can be avoided. Therefore, in our approach, we will define a probability distribution over all possible segmentations of the scene to deal with noise and uncertainty. We will show that such an approach can be effectively used for scene segmentation.

2.2.4 Combining Interaction and Visual Clues

Interaction can yield powerful clues, but might take a substantial amount of time. In contrast, visual segmentation clues may be ambiguous but are instantly available. Combining both kinds of clues can potentially reduce the interaction time needed to obtain good segmentations.

For example, Katz et al. [2010] used hand-tuned predictors based on compactness and appearance, in addition to co-movement of features. Bergström et al. [2011] combined rigid motion clues with color- and disparity clues, whereas Schiebener et al. [2013] validated hypotheses based on proximity and shared parametric surfaces using co-movement. Hausman et al. [2013] used visual features to generate hypotheses, and to reconstruct a dense model from clustered feature points.

The methods discussed in the previous paragraph used visual features to create hypotheses that were tested by interaction [Bergström et al., 2011, Schiebener et al., 2013, Hausman et al., 2013] or to modify binary potentials between parts [Katz et al., 2010]. Instead, a *probabilistic approach* offers a principled way to integrate noisy clues from multiple sources. Likelihood terms based on different clues can than be integrated in a principled manner. Learning the parameters of the likelihood model from past experience enables knowledge transfer between different scenes and avoids hand-tuning.

In this chapter, we will consequently extend the approach proposed in Sec. 2.2.3 with factors corresponding to static scene properties detected by the visual system. We will show that this information speeds up the segmentation process substantially.

2.2.5 Efficient Learning with Informed Exploration

In cluttered environments, many different explorative actions are possible. Not all of these actions are equally informative, so that carefully selecting informative actions may decrease the amount of interaction time needed. Surprisingly, all methods discussed in Sec. 2.2.2 employed actions that were selected at random, or according to fixed schemes and heuristics. Fixed actions were applied in scenarios with only a single object, and hence such approaches cannot deal with cluttered environments. Heuristics rely on human insights into the problem domain, and as such are not at all

guaranteed to work in different domains or unforeseen situations. An exception is the approach of Hausman et al. [2013], where actions are chosen based on the probability that the proposed object is the result of over- or under-segmentation. This probability is determined based only on the number of segments assigned to that object, and does not take the likelihood of plausible alternatives into account.

Rather than relying on human cleverness to tweak heuristics, it would be preferable to use general principles that allow the robot to adapt to new situations autonomously. Principles from information theory, for example, can be used to quantify the informativeness of possible actions. For example, perceptual parameters can be chosen to maximize the informativeness of observations [Huber et al., 2012, Denzler and Brown, 2002, Schneider et al., 2009, Hsiao et al., 2011], often after prior training on a specific set of objects or using a physics simulator.

There are several examples of information-theoretic approaches to interactive perception. For example, Krainin et al. employed a next-best-view algorithm based on information gain to select the best viewpoint for in-hand object modeling [Krainin et al., 2011a]. This approach assumes the robot knows the objects in its environment sufficiently well to grasp them. Similarly, Sushkov and Sammut [Sushkov and Sammut, 2012] selected interactions using the expected information gain, based on discrete sets of possible actions and models provided by a human and training in a physics simulator.

Informative actions can also be found by selecting actions that were successful in uncovering new object properties in similar situations in the past. Katz et al. [2008], for example, estimated the value of actions for exploring the kinematic structure of articulated objects using Q-learning. Their approach used a human-crafted, domain-specific representation to generalize past experiences to the current situation.

In our approach, knowledge of the uncertainty of the scene's segmentation is captured in a probabilistic model. We can exploit this knowledge to calculate the (approximate) expected information gain of possible actions, without additional hand-tuning. We will use this criterion as a principled way to select informative explorative actions, and show that using such actions improves performance in an example task.

2.3 Probabilistic Segmentation and Modeling

In environments cluttered with novel objects, the correct segmentation of the scene into its constituent objects is initially unknown. Hence, appearance characteristics cannot be attributed to the correct object, posing challenges to tracking. We rather attribute appearance characteristic and movement to local regions called 'parts'. This part-based approach allows the resulting motion of an action to be estimated at the object level, which prevents problems associated with estimating such movement at a pixel level [Beale et al., 2011]. In Sec. 2.3.1 and 2.3.2, our approach to obtaining and tracking such regions will be explained. By using the part-based model, the segmentation problem is reduced to finding out how the parts are grouped into objects. How this partitioning is learned by interacting with objects is detailed in Sec. 2.3.3 and 2.4.4. Finally, Sec. 2.3.5 describes how actions can be selected to maximize the expected information gain. Figure 2.2 illustrates this process.

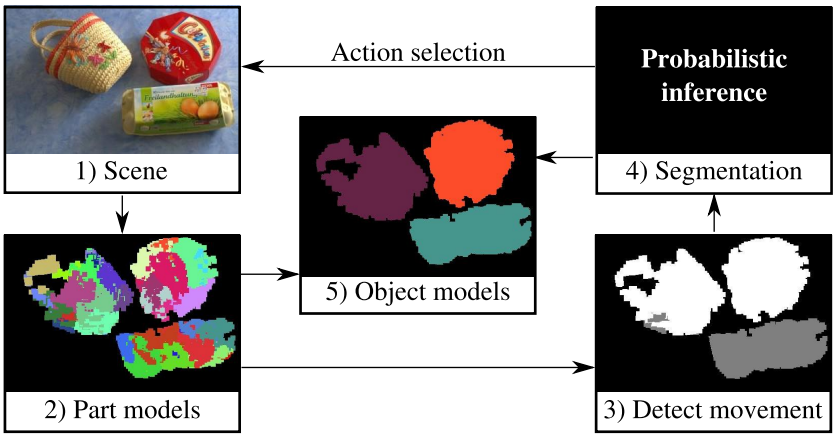


Figure 2.2.: Every scene is processed using our part-based approach: (1,2) Known parts are recognized using stored appearance characteristics and new parts are extracted from regions that have not been seen yet. (3) Movement of known parts (shown in gray) is detected based on the distance between their current and past locations. (4) By merging part models according to the inferred partitioning, a partitioning of parts is determined, corresponding to a segmentation of the scene.

2.3.1 Part Extraction and Description

We initialize parts using a three-dimensional grid covering the observed point cloud. Each of these grid points ('center points') defines a part consisting of all points within a fixed radius of these center points. Parts may overlap each other. As objects are pushed, previously obscured points of the object surface can become visible. If such points are not within the radius of existing parts, a new part center can be generated at its location. If the radius is set too large, a single part could span multiple objects. On the other hand, setting it too small results in unnecessarily many parts, driving up computational requirements. A radius of 6 cm worked sufficiently well in our experiments, as this radius corresponds roughly to the smallest dimension of our objects.

The parts are tracked using local key point descriptors within their radius. Key points are a sparse set of points that can be detected reliably from multiple views. An expressive description makes sure that correspondences between key points extracted from different views can be matched reliably. As a sparse set of key points is used, calculating and matching descriptors can be done relatively fast.

Our approach does not depend on the particular kind of descriptors employed. In our experiments, key points and descriptors were obtained using the Scale Invariant Feature Transform algorithm [Lowe, 2004]. We considered to add Maximally Stable Color Regions [Forssén, 2007], as in [Schiebener et al., 2013]. However, we found that adding these features did not make a practical difference in tracking for our set of objects (Sec. 2.4.1), as the observed median difference in part locations was less than 1 cm.

2.3.2 Part Recognition and Movement Detection

Our approach requires the detection of the movement of each of the parts resulting from actions taken by the robot. The inference of the scene segmentation is independent of the method employed to detect these movement. We employed an 'eye in hand' setup, and compare the locations of key points [Lowe, 2004] before and after a push to detect part motion.

Occasionally, false matches occur, even with powerful descriptors. Although we do not assume *objects* are rigid, we do approximate the movement of the object's *parts* by homogeneous transformations, which helps to filter out false matches. We use the random sample consensus (RANSAC) algorithm [Fischler and Bolles, 1981] to robustly find the rigid 3D transformation of the local coordinate frame that explains most key-point matches. In each iteration, a homogeneous transform is fitted using singular value decomposition. In our experiments, we used 100 iterations of the algorithm.

The transformation with the highest number of inliers is accepted if it exceeds a fixed threshold (in our experiments, a quarter of all matches with a minimum of six). We refit he transformation using all found inliers for stability. The refitted transformation is accepted if it has a higher number of inliers.

If the found transformation includes a translation of more than a threshold of three cm (i.e., half of the distance that the robot tries to push the object), we conclude the part has moved. We considered adding a criterion based on pure rotations, but this did not improve the performance of our method. If no rigid transformation with sufficient inliers is found, we conclude that the part is not visible in the scene, for example due to occlusion of the object.

In our current setup, we are limited to low-frequency change detection instead of continuous tracking because of the minimum distance to the scene (80 cm) required by our sensor. In a hardware setup with an independently positioned sensor, continuous tracking might yield more robust results, and could deal with textureless objects as well [Yilmaz et al., 2006].

2.3.3 Interaction for Probabilistic Segmentation

To obtain a scene segmentation, the extracted object parts have to be partitioned into groups corresponding to the objects in the scene. Clues for this partitioning are provided by interacting with the environment, which results in the movement of objects. Observing this movement can resolve segmentation ambiguities [Kenney et al., 2009, Fitzpatrick and Metta, 2003].

Seeing parts move together is more probable when these parts belong to the same objects. However, observed joint movement does not guarantee that features belong to the same object, as the robot's sensors are noisy and objects that push each other also result in joint movement. Therefore, we retain a distribution over possible segmentations, rather than choosing the most likely segmentation. As the movement resulting from consecutive actions is observed, this distribution will generally become

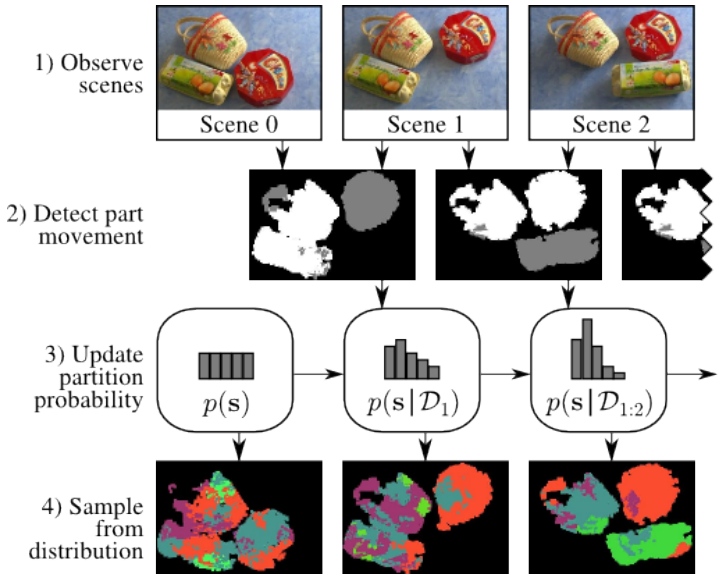


Figure 2.3.: Overview of our probabilistic segmentation approach. After every action, the resulting scene is observed (1) and moving (gray) and non-moving (white) parts are identified (2). This data set $\mathcal{D}$ is subsequently used to update the probability distribution over partitions $\mathbf{s}$ (3). This distribution is approximated using samples (4).

narrower, and will have more probability mass at the true segmentation. This process is illustrated in Figure 2.3.

Graphical Model and Notation

We represent a segmentation by a vector $\mathbf{s} = [s_1, \ldots, s_N]^T$, with the elements s_i indicating the object containing each of the N parts. For example, the vector $\mathbf{s} = [1, 1, 2]^T$ would indicate a segmentation where the first two parts are assigned to the same object, while the third part is assigned to a different object.

To infer a scene's segmentation the robot has access only to data set $\mathcal{D}_T = \{a_t, \mathbf{o}_t | t \leq T\}$, where a_t is the index of the pushed part at time t and $\mathbf{o}_t$ the resulting observation vector. Its j^{th} value $\mathbf{o}_t[j]$ is equal to 1 if part j was observed moving at time step t, or 0 if it was observed to be still.

We assume that the probability that an object moves is an unknown constant θ_{p} if pushed, or θ_{np} if not pushed. The movement of part n at time t is represented with a latent binary variable $m_{n,t}$. This variable $m_{n,t}$ has the same value for all parts belonging to the same object: $s_n = s_j \implies m_{n,t} = m_{j,t}$. Part n is as-

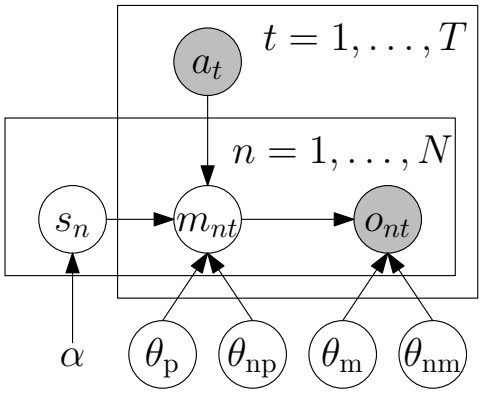


Figure 2.4.: Graphical model for probabilistic segmentation. Shown are parameters, hidden variables (open circles) and observed variables (shaded circles).

(a) Test scene to be segmented; true segmentation.

Prior (0 actions) | Posterior (5 actions) | Posterior (15 actions)

(b) Samples of the distribution over segmentations s, coloring the parts according to the object they are assigned to. The columns contain three independent samples from the distribution after observing the effect of 0, 5 and 15 actions, respectively.

Figure 2.5.: Samples of the distribution over segmentations of a test scene. A priori, the number of objects as well as the segmentation are unknown. Therefore, samples of the prior are very different from each other. Over time, the number of objects and the correct segmentation are inferred. The growing consistency of the samples indicates decreasing uncertainty.

sumed to be observed moving at time t with an unknown, fixed probability θ_{m} if the object to which it belongs moved ($m_{n,t} = 1$) or θ_{nm} if it did not. The corresponding graphical model is shown in Figure 2.4.

Sampling approach for segmentation inference

The number of possible segmentations grows exponentially as the number of parts increases. Hence, calculating the probability of each segmentation quickly becomes computationally intractable. Nevertheless, we can approximate this distribution using samples, which can be drawn using Markov chain Monte Carlo methods such as the Gibbs sampler [Geman and Geman, 1984]. Given data set $\mathcal{D}_T$, this approach produces samples from a joint distribution over latent variables by iteratively selecting one latent variable, and assigning it a value according to the assignment probability conditioned on the assignment of all other sampled variables. If we keep only every k^{th} sample, we obtain independent samples for sufficiently large k.

The parameters $\theta_{\mathrm{p}}, \theta_{\mathrm{np}}, \theta_{\mathrm{m}}$ and θ_{nm} can be marginalized in closed form for conjugate priors. Elements s_i of $\mathbf{s}$ and $m_{n,t}$ of $\mathbf{m}$ are sampled according to proposal

distributions depending on the current sampled values of all other elements of those vectors, denoted by $s_{\setminus i}$ and $m_{\setminus n,t}$:

$$p(s_i|s_{\setminus i}, \mathbf{a}, \mathbf{m}, \alpha) \propto p(\mathbf{m}|\mathbf{a}, \mathbf{s}) p(s_i|s_{\setminus i}, \alpha), \text{ and} \tag{2.1}$$

$$\begin{aligned} p(m_{n,t}|\mathcal{D}_T, \mathbf{s}, m_{\setminus n,t}) &\propto p(\mathbf{m}|\mathcal{D}_T, \mathbf{s}) \\ &\propto p(\mathbf{o}|\mathbf{m}) p(\mathbf{m}|\mathbf{a}, \mathbf{s}), \end{aligned} \tag{2.2}$$

exploiting the conditional independences expressed in the graphical model (Figure 2.4). The following subsections will explain how we define the prior $p(s_i|s_{\setminus i}, \alpha)$, the movement model $p(\mathbf{m}|\mathbf{a}, \mathbf{s})$ and the observation model $p(\mathbf{o}|\mathbf{m})$ needed to evaluate these expressions. Examples of samples drawn using this approach are shown in Figure 2.5.

Prior Distribution over Segmentations. The number of objects in the scene is not assumed to be known. Hence, a suitable non-parametric prior distribution $p(s_i|s_{\setminus i})$ over partitionings $\mathbf{s}$ of n parts is given by the Chinese restaurant process [Aldous, 1985]. Given the assignment of the other parts $s_{\setminus i}$, part i is assigned to an existing object j with a probability dependent on the number of parts n_j already assigned to that object: $p(s_i = j|s_{\setminus i}) = n_j/(\alpha + n - 1)$. However, with probability $p(s_i = J|s_{\setminus i}) = \alpha/(\alpha + n - 1)$ the part can also be assigned to a new object J, to which no other part has been assigned yet. Due to the non-parametric nature of the Chinese restaurant process, the number of objects does not need to be set in advance, but is learned from the data. The free parameter α controls how often new objects are created by the generative process. In our experiments, α was set to 1.

The Movement Model. To evaluate Equations (2.1) and (2.2), we define a movement model

$$\begin{aligned} p(\mathbf{m}|\mathbf{a}, \mathbf{s}) &= \int_0^1 \int_0^1 p(\mathbf{m}|\mathbf{s}, \theta_{\mathrm{p}}, \theta_{\mathrm{np}}) p(\theta_{\mathrm{p}}, \theta_{\mathrm{np}}|\mathbf{s}) \mathrm{d}\theta_{\mathrm{p}} \mathrm{d}\theta_{\mathrm{np}} \\ &\propto \int_0^1 \theta_{\mathrm{p}}^{\alpha'_{\mathrm{p}}} \left(1 - \theta_{\mathrm{p}}\right)^{\beta'_{\mathrm{p}}} \mathrm{d}\theta_{\mathrm{p}} \int_0^1 \theta_{\mathrm{np}}^{\alpha'_{\mathrm{np}}} \left(1 - \theta_{\mathrm{np}}\right)^{\beta'_{\mathrm{np}}} \mathrm{d}\theta_{\mathrm{np}} \\ &\propto \frac{\alpha'_{\mathrm{p}}! \beta'_{\mathrm{p}}!}{\left(1 + \alpha'_{\mathrm{p}} + \beta'_{\mathrm{p}}\right)!} \frac{\alpha'_{\mathrm{np}}! \beta'_{\mathrm{np}}!}{\left(1 + \alpha'_{\mathrm{np}} + \beta'_{\mathrm{np}}\right)!}, \end{aligned}$$

with factorizing conjugate prior

$$p(\theta_{\mathrm{p}}, \theta_{\mathrm{np}}|\mathbf{s}) = \mathrm{Beta}(\theta_{\mathrm{p}}|\alpha_{\mathrm{p}}, \beta_{\mathrm{p}}) \mathrm{Beta}(\theta_{\mathrm{np}}|\alpha_{\mathrm{np}}, \beta_{\mathrm{np}}),$$

and α'_{p}, α'_{np} are the corresponding α_{p}, α_{np} plus the number of times an object moved given that it was pushed (α'_{p}) or not pushed (α'_{np}). Similarly, β'_{p}, β'_{np} are the corresponding β_{p}, β_{np} plus the number of times an object did not move given that it was pushed (β'_{p}) or not pushed (β'_{np}). We set the hyper-parameters $\alpha_{\mathrm{p}} = \beta_{\mathrm{np}} = \alpha_{\mathrm{np}} = \beta_{\mathrm{p}} = 1$ to encode a uniform prior.

The Observation Model. Analogous to the movement model, we define the observation model

$$\begin{aligned} p(\mathbf{o}|\mathbf{m}) &= \int_0^1 \int_0^1 p(\mathbf{o}|\mathbf{m}, \theta_{\mathrm{m}}, \theta_{\mathrm{nm}}) p(\theta_{\mathrm{m}}, \theta_{\mathrm{nm}}) \mathrm{d}\theta_{\mathrm{m}} \mathrm{d}\theta_{\mathrm{nm}} \\ &\propto \int_0^1 \theta_{\mathrm{m}}^{\alpha'_{\mathrm{m}}} (1-\theta_{\mathrm{m}})^{\beta'_{\mathrm{m}}} \mathrm{d}\theta_{\mathrm{m}} \int_0^1 \theta_{\mathrm{nm}}^{\alpha'_{\mathrm{nm}}} (1-\theta_{\mathrm{nm}})^{\beta'_{\mathrm{nm}}} \mathrm{d}\theta_{\mathrm{nm}} \\ &\propto \frac{\alpha'_{\mathrm{m}}! \beta'_{\mathrm{m}}!}{\left(1+\alpha'_{\mathrm{m}}+\beta'_{\mathrm{m}}\right)!} \frac{\alpha'_{\mathrm{nm}}! \beta'_{\mathrm{nm}}!}{\left(1+\alpha'_{\mathrm{nm}}+\beta'_{\mathrm{nm}}\right)!}, \end{aligned}$$

with factorizing conjugate prior

$$p(\theta_{\mathrm{m}}, \theta_{\mathrm{nm}}) = \mathrm{Beta}(\theta_{\mathrm{m}}|\alpha_{\mathrm{m}}, \beta_{\mathrm{m}}) \mathrm{Beta}(\theta_{\mathrm{nm}}|\alpha_{\mathrm{nm}}, \beta_{\mathrm{nm}}).$$

The variables α'_{m}, α'_{nm}, β'_{m}, β'_{nm} express the number of parts that were observed moving or not moving given movement of the corresponding object, added to the respective hyper-parameters $\alpha_{\mathrm{m}} = \beta_{\mathrm{nm}} = \alpha_{\mathrm{nm}} = \beta_{\mathrm{m}} = 1$. In subsequent sections, we will need the data likelihood

$$p(\mathcal{D}_T|\mathbf{s}) = \mathbb{E}_{\mathbf{m}}[p(\mathcal{D}_T|\mathbf{m})|\mathbf{s}] \approx J^{-1} \sum_{j=1}^{J} p(\mathcal{D}_T|\mathbf{m}_j),$$

using the conditional independence of $\mathcal{D}_T$ from $\mathbf{s}$ and approximating the expectation with samples $\mathbf{m}_j \sim p(\mathbf{m}|\mathbf{s})$. Furthermore, we will need the observation probability

$$p(\mathbf{o}|a, \mathbf{s}, \mathcal{D}_T) = \mathbb{E}_{\mathbf{m}}[\, p(\mathbf{o}|\mathbf{m})|\, a, \mathbf{s}, \mathcal{D}_T\,] \approx J^{-1} \sum_{j=1}^{J} p(\mathcal{D}_T|\mathbf{m}_j),$$

using the conditional independence of $\mathbf{o}$ and samples $\mathbf{m}_j \sim p(\mathbf{m}_j|a, \mathbf{s}, \mathcal{D}_T)$.

2.3.4 Integrating Visual Clues

In case we have access to vision data set $\mathcal{D}_{\mathrm{vis}}$ besides interaction data set $\mathcal{D}_T$, it is straightforward to adapt our model

$$p(\mathbf{s}|\mathcal{D}_T, \mathcal{D}_{\mathrm{vis}}) \propto p(\mathcal{D}_T|\mathbf{s}) p(\mathcal{D}_{\mathrm{vis}}|\mathbf{s}) p(\mathbf{s}),$$

under the assumption that data sets $\mathcal{D}_T$ and $\mathcal{D}_{\mathrm{vis}}$ are conditionally independent (see Figure 2.6). One important clue obtained from vision is the spatial distribution of the parts in the scene. If we assume objects to be spatially compact, the locations $\{\mathbf{x}_j|s_j = k\}$ of parts object k are clustered around some location $\boldsymbol{\phi}_k$. We assume $p(\mathbf{x}_j|\boldsymbol{\phi}_{s_j}) = \mathcal{N}(\mathbf{x}_j; \boldsymbol{\mu}_j, \Sigma)$ to be a normal distribution with $\boldsymbol{\mu}_j = \boldsymbol{\phi}_{s_j}$. The variables $\boldsymbol{\phi}_i$ are latent; a conjugate prior over latent variables $\boldsymbol{\phi}_k$:$p(\boldsymbol{\phi}_k) = \mathcal{N}(\boldsymbol{\phi}_k; \mathbf{0}, V)$ allows marginalization in closed form. The desired likelihood

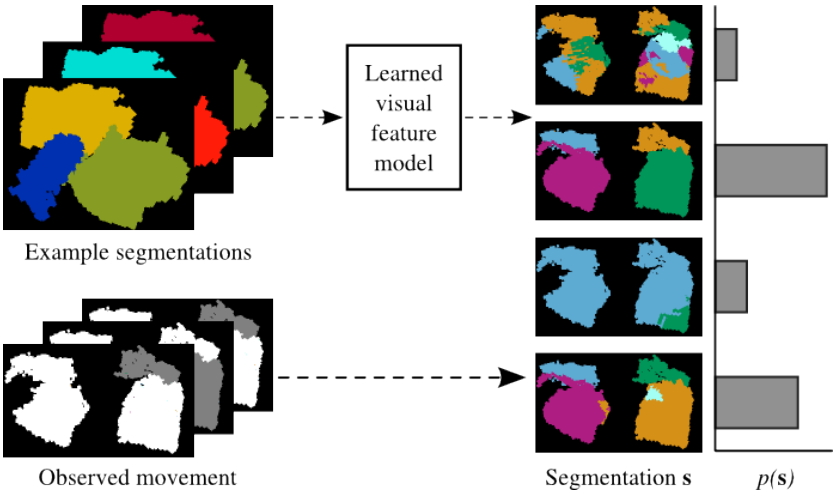


Figure 2.6.: Statistical properties of visual features in previously-seen scenes can be modeled. The visual likelihood of segmentations of a new scene is combined with movement likelihood to improve calculated segmentation probabilities.

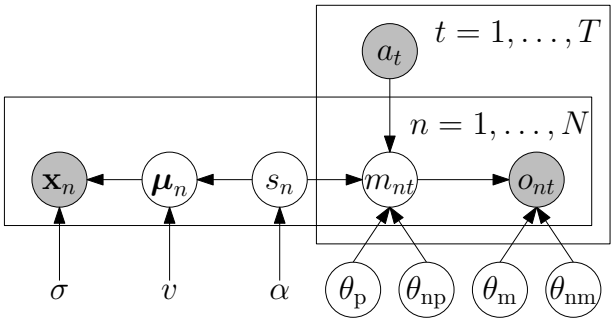


Figure 2.7.: Graphical representation of the extended model. The earlier model in Figure 2.4 is extended by the centers of object $\boldsymbol{\mu}$ and the observed centers of parts $\mathbf{x}$.

$$p(\mathcal{D}_{\mathrm{vis}}|\mathbf{s}) = \prod_{k \in \mathcal{K}} \left(\iint_{\mathbb{R}^2} p(\boldsymbol{\phi}_k) \prod_{j \in \mathcal{J}_k} p(x_j|\boldsymbol{\phi}_{s_j}) \mathrm{d}\boldsymbol{\phi}_k \right),$$

where $\mathcal{K}$ is the set of objects in the current segmentation and $\mathcal{J}_k = \{j | 1 \leq j \leq N, s_j = k\}$ is the set of indexes of parts that belong to object k. The extended model is shown in Figure 2.7.

For simplicity, we assume the normal distributions to be isotropic ($\Sigma = \sigma I$, $V = vI$) and train the parameters σ and v on a data set of already-segmented scenes using the method of maximal marginal likelihood using a gradient ascent approach. No additional tuning is required.

Our modeling approach is quite general: different kinds of features (e.g., based on color or shape) can be integrated in the model in the same manner. Only a suitable distribution over the properties of the objects and their parts is required. We chose the location feature due to its generality, as spatial compactness holds for a wide variety of objects and scenes.

2.3.5 Maximizing Mutual Information for Directed Exploration

Our approach generates part models and approximates a distribution over segmentations regardless of the chosen actions. However, if our robot deliberately chooses informative actions, we expect useful object models to be learned faster [Denzler and Brown, 2002, Schneider et al., 2009, Hsiao et al., 2011, van Hoof et al., 2012]. Hence, we choose actions to maximize the mutual information $I(\mathbf{s};\mathbf{o}|a,\mathcal{D}_T)$, where $\mathbf{s}$ is the partition of the parts into objects and $\mathbf{o}$ is the observed outcome of the action targeted at part a at $t = T+1$. The mutual information is equal to the expected information gain of observing the result of pushing part a given by

$$\begin{aligned} I(\mathbf{s};\mathbf{o}|a,\mathcal{D}_T) &= \mathbb{E}_{\mathbf{o}}\left[D_{\mathrm{KL}}(p(\mathbf{s}|\mathbf{o},a,\mathcal{D}_T)||p(\mathbf{s}|\mathcal{D}_T))|a,\mathcal{D}_T\right] \\ &= \mathbb{E}_{\mathbf{s},\mathbf{o}}\left[\log\left(\frac{p(\mathbf{s},\mathbf{o}|a,\mathcal{D}_T)}{p(\mathbf{o}|a,\mathcal{D}_T)p(\mathbf{s}|a,\mathcal{D}_T)}\right)\middle| a,\mathcal{D}_T\right], \end{aligned}$$

where D_{KL} is the Kullback-Leibler divergence. The argument of the logarithm is computed using

$$\begin{aligned} \frac{p(\mathbf{s},\mathbf{o}|a,\mathcal{D}_T)}{p(\mathbf{o}|a,\mathcal{D}_T)p(\mathbf{s}|a,\mathcal{D}_T)} &= \frac{p(\mathbf{o}|\mathbf{s},a,\mathcal{D}_T)p(\mathbf{s}|a,\mathcal{D}_T)}{p(\mathbf{o}|a,\mathcal{D}_T)p(\mathbf{s}|a,\mathcal{D}_T)} \\ &= \frac{p(\mathbf{o}|\mathbf{s},a,\mathcal{D}_T)}{\mathbb{E}_{s'}\left[p(\mathbf{o}|\mathbf{s}',a,\mathcal{D}_T)|\mathcal{D}_T\right]}, \end{aligned}$$

as $p(\mathbf{s}|a,\mathcal{D}_T) = p(\mathbf{s}|\mathcal{D}_T)$. The spaces $\mathcal{S}$ and $\mathcal{O}$ of possible partitions and observations grow exponentially as the number of parts increases. Hence, evaluating these expectations exactly is infeasible. We can approximate these expectations using samples $(\mathbf{s}_{(j)},\mathbf{o}_{(j)}) \sim p(\mathbf{s},\mathbf{o}|a,\mathcal{D}_T)$, $j \in \{1,\ldots,J\}$ and samples $\mathbf{s}_{(k)} \sim p(\mathbf{s}|\mathcal{D}_T)$, $k \in \{1,\ldots,K\}$, i.e., by computing

$$I(\mathbf{s};\mathbf{o}|a,\mathcal{D}_T) \approx \frac{1}{J}\sum_j \log\left(\frac{p(\mathbf{o}_{(j)}|\mathbf{s}_{(j)},a,\mathcal{D}_T)K}{\sum_k p(\mathbf{o}_{(j)}|\mathbf{s}_{(k)},a,\mathcal{D}_T)}\right). \tag{2.3}$$

The samples from $p(\mathbf{s}|\mathcal{D}_T)$ can be obtained using the Gibbs sampler described in Sec. 2.3.3. Now, we sample from the conditional $p(\mathbf{o}|\mathbf{s},a,\mathcal{D}_T)$ to get a sample from $p(\mathbf{o},\mathbf{s}|a,\mathcal{D}_T)$. Furthermore, we need the conditional observation probability

$$p(\mathbf{o}|a,\mathbf{s},\mathcal{D}_T) = \frac{p(\mathbf{o},\mathcal{D}_T|\mathbf{s},a)}{p(\mathcal{D}_T|\mathbf{s})},$$

as $p(\mathcal{D}_T|\mathbf{s},a) = p(\mathcal{D}_T|\mathbf{s})$. We determine $p(\mathcal{D}_T|\mathbf{s})$ as described in Sec. 2.3.3, and compute $p(\mathbf{o},\mathcal{D}_T|\mathbf{s},a)$ similarly after adding the potential action a and observation $\mathbf{o}$ to the actual actions and observations in $\mathcal{D}_T$.

2.4 Evaluation

In the proposed approach, a robot uses probabilistic inference to segment a cluttered scene based on interaction data. In this section, we will first introduce our general

experimental setup in Sec. 2.4.1. Subsequently, in Sec. 2.4.2 we compare our probabilistic segmentation method to alternative approaches on data gathered by a real robot. In Sec. 2.4.3, we consider a scenario where action selection according to the mutual information criterion is needed to learn efficiently, and compare that strategy to random action selection. Finally, we evaluate the inclusion of a likelihood term based on the visually observed spatial distribution of object parts in addition to interaction data in Sec. 2.4.4.

2.4.1 Experimental Setup and Quality Measure

We evaluated our approach using a 7 degrees of freedom Mitsubishi PA-10 robot arm. A RGBD camera, a force-torque sensor, and a rod used to manipulate the objects were mounted on the arm's end effector (see Figure 2.1). Hence, the robot could move the camera to observe the scene from different perspectives. The force-torque sensor allowed the robot to register forces exerted on the rod, which allowed the robot to autonomously stop its motion in case of unexpected collisions. The camera was calibrated. Consequently, observations taken from different points of view could be aligned straightforwardly by transforming them to the robot's coordinate frame. Since the table location is known, observed parts not belonging to the scene on the table could automatically be removed.

The robot was presented with a cluttered scene of novel objects taken from the set shown in Figure 2.8. These objects were set up on a table next to the robot. The robot interacted with the scene, pushing the selected part near its center in a direction based on its estimated surface normal. After every action, the scene was observed from three different view points in order to update the distribution over partitions of the parts into objects (Sec. 2.3.3). The entire procedure is illustrated in Figure 2.9. Performing this procedure took about one minute for each action, most of which was used by actual movement of the robot. Our robot was not moving at maximum speed to keep operation controlled and safe.[1]

Figure 2.8.: The set of 12 everyday objects used in our experiments. We included objects of different shapes and an articulated object (train), a deformable object (cloth bundle) and a flexible object (basket).

Quality Measure. After every action, the robot updates its posterior probability distribution over segmentations. Parts that belong to the same object according to the ground truth (human annotation), should be assigned likewise by the robot. The robot decides the parts should belong to the same object if the majority of samples assigns them so (and vice versa).

[1] A video showing our setup is available at: http://youtu.be/GQYP2eYaGks.

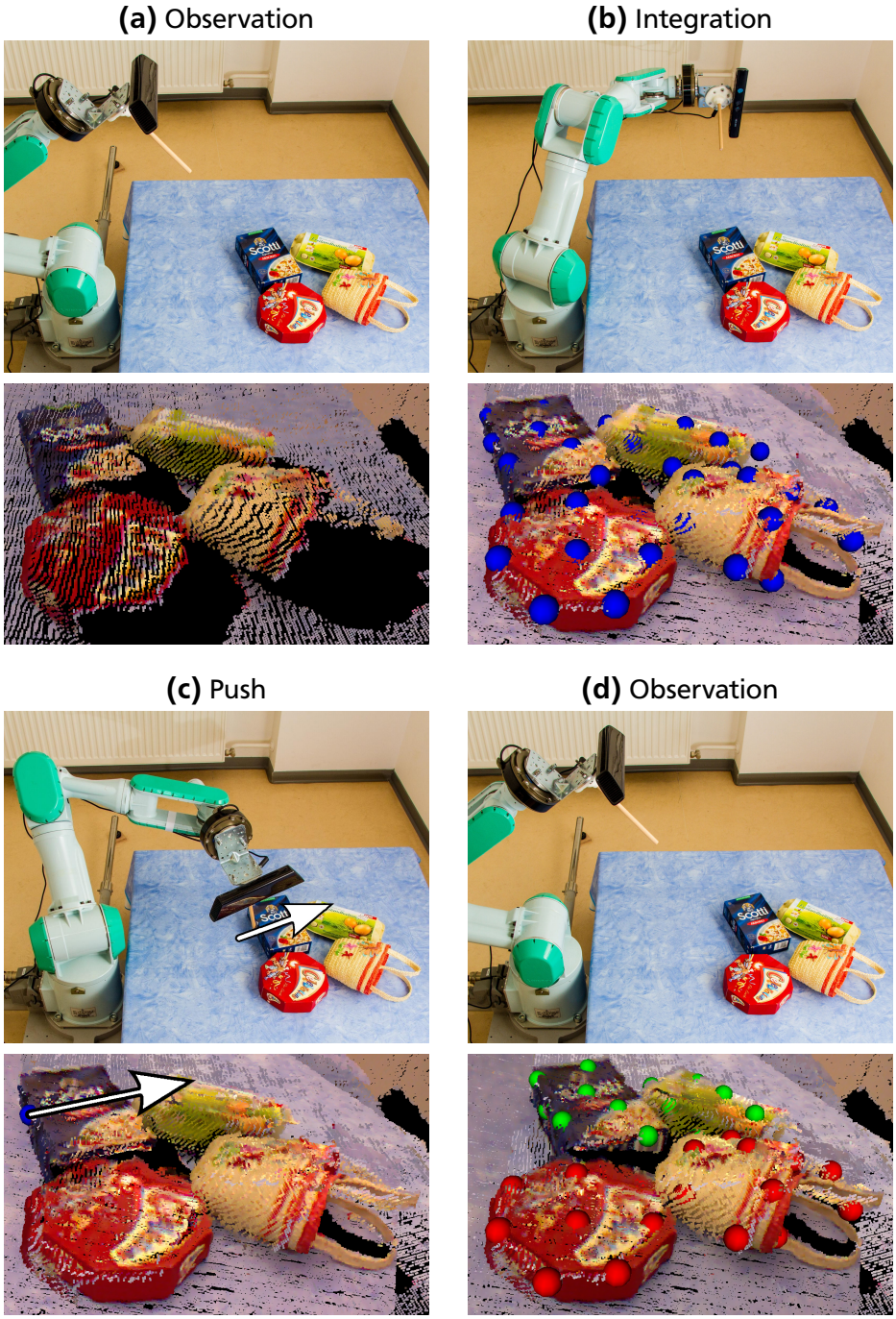


Figure 2.9.: Illustration of the exploration phase. (a) The robot observes the scene, obtaining an incomplete point cloud from one perspective. (b) Percepts from multiple perspectives are integrated and patches are extracted (part centers shown as blue spheres). (c) A push is selected (bottom, blue sphere) and executed. (d) The resulting scene is observed and the patch centers are registered as moving (green) or non-moving (red).

Following Fowlkes and Mallows [1983], the quality measure we used was the correspondence

$$B = \frac{|P \cap Q|}{\sqrt{|P||Q|}}, \tag{2.4}$$

where Q and P are the set of pairs of parts that belong to the same object according to the human annotation (Q) or according to the model's prediction (P), and $|\cdot|$ denotes a set's cardinality. This correspondence is zero if ground truth and prediction do not agree on any pair of parts. Conversely, the correspondence is one if they agree on all pairs.

2.4.2 Interactive Segmentation Experiment

During interaction with its environment, the robot obtained information that allowed it to narrow down its distribution over possible segmentations. The number of objects and the way the parts should be assigned to those objects were inferred simultaneously with the parameters $\theta_{\mathrm{p}}, \theta_{\mathrm{np}}, \theta_{\mathrm{m}}$ and θ_{nm}.

We compared our probabilistic model to two heuristics commonly employed in the field of interactive segmentation. Neither of these baselines directly re-implements a particular approach from the literature: interactive segmentation approaches usually have a strong interdependency between setup, sensing, representation, inference and action selection, making it difficult to execute such a direct comparison in a fair manner. These baselines are:

Rigid Motion Approach. If parts do not follow the same rigid transformation, they need to belong to different objects.

Pairwise Approach. Two parts belong together if co-movement is observed more often than separate movement, independently from any other parts.

The set of twelve different objects (Figure 2.8) was used to create fifteen initial setups consisting of four objects each. The robot explored these objects using fifteen random actions. Only motion data was used in this experiment. The same data set was used for all methods.

Discussion. In the segmentation task, we evaluated the quality of the segmentation the robot inferred through interaction. The robot learned continuously from its own experience, without an annotated training set or reward signals.

The experiment required inferring the segmentation of scenes composed of rigid and non-rigid objects using just movement data. Considering these circumstances, both the pairwise method and our full model did quite well. Both comparison methods were outperformed by our probabilistic segmentation approach (see Figures 2.10 and 2.11). Our approach attained an average quality of 0.86 in contrast to 0.80 for the pairwise method. This difference is a relevant step towards perfect segmentations (1.00). The rigid motion method is sensitive to tracker errors, and attained an average quality of only 0.52. After 15 actions all methods except for the rigid motion method

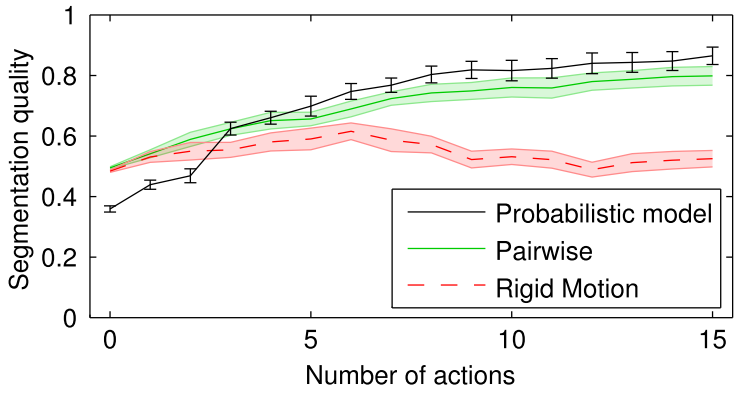


Figure 2.10.: Inference of scene segmentations using different inference methods. Parameter learning increases initial uncertainty in the probabilistic model, which reduces segmentation quality initially. The error bars and shaded areas show the standard error.

appear to have converged. Our probabilistic approach needed only nine actions for half of the trial runs to attain a segmentation quality of at least 0.85, while this quality was not reached within 15 actions for the pairwise and rigid motion methods (see Figures 2.10 and 2.11).

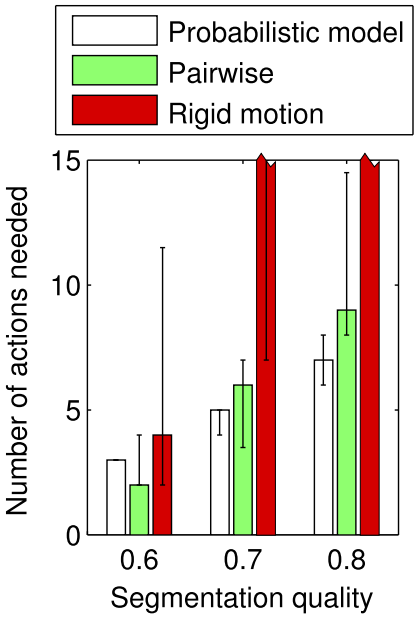


Figure 2.11.: Scene segmentations using different inference methods. Bars show the median number of actions needed, error bars indicate the interquartile range where it is different from the median. Trials were cut off after 15 actions.

Our scenario is different from that of related work in terms of the available knowledge. Related work often focused on scenario's where knowledge of the objects' movement type (e.g., rigid transformation) were given to the robot [Bergström et al., 2011, Hermans et al., 2012, Cho et al., 2008, Hausman et al., 2013, Beale et al., 2011], whereas in our case only the occurrence of movement was used. This additional flexibility has the cost that we usually need more actions than these related approaches. For example, parameters of our probabilistic model ($\theta_{\mathrm{p}}, \theta_{\mathrm{np}}, \theta_{\mathrm{m}}$ and θ_{nm}) are learned rather than tuned which increases uncertainty in the beginning of the experiment, reducing performance of our learned model relative to manually specified models initially. The baseline methods, on the other hand, do not optimally exploit the information in larger data sets. One cause for occasional failure in all of the methods was that the tracker system occasionally loses certain parts (see Sec. 2.4.3).

A qualitative advantage of our method over the pairwise method is that it respects the transitivity of the 'belongs to the same object' relation: if part

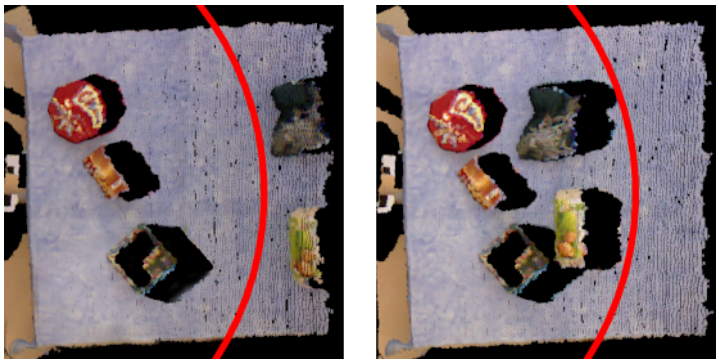


Figure 2.12.: Top-view of the setup, with the robot located outside of the illustration on the left-hand side. In the initial setup, two objects are visible to the robot but outside of its workspace (indicated by the red arc). After the tenth action (left), those objects are moved manually within the robot's workspace (right). This movement is not used in inference. After observing the resulting situation, the robot continues with the eleventh action.

i and j belong together and so do j and k, the same necessarily holds for i and k. The pairwise method cannot guarantee this consistency.

2.4.3 Action Selection Experiment

In a second experiment, we evaluate how much the robot gains by exploiting its knowledge of the segmentation uncertainty to select more informative actions. When all objects can be manipulated equally easily, random action selection performs fairly well as it tends to distribute actions evenly over all objects. However, objects are not always consistently reachable. Some objects might even be entirely out of the robot's workspace, and can only be manipulated later.

Therefore, we used a setup similar to the previous experiment, however, every scene included five objects of which two were initially placed outside of the robot's workspace. The objects were selected from the set shown in Figure 2.8, and ten independent trials were executed. After training on the three remaining objects for ten actions (using a random action selection strategy), these objects were placed such that they became reachable to the robot (Figure 2.12).

Subsequently, five more actions were executed using one of two action selection strategies: selecting actions at random or according to the maximal mutual information criterion (Sec. 2.3.5). To speed up computation, we set the parameters $\theta_{\text{np}}, \theta_{\text{p}}, \theta_{\text{m}}, \theta_{\text{nm}}$ to their MAP estimate using the data gathered in the previous experiment. Action selection took less than one second. The action's resulting movement was used to infer the segmentation using our probabilistic approach.

Discussion. The results of the action selection experiments are shown in Figure 2.13. After the first ten actions, three objects initially in the workspace had been explored using random actions. When two initially out-of-reach objects were introduced into the workspace, random actions were divided over all five objects, with a bias toward objects that were already in the robot's workspace. This bias might occur because

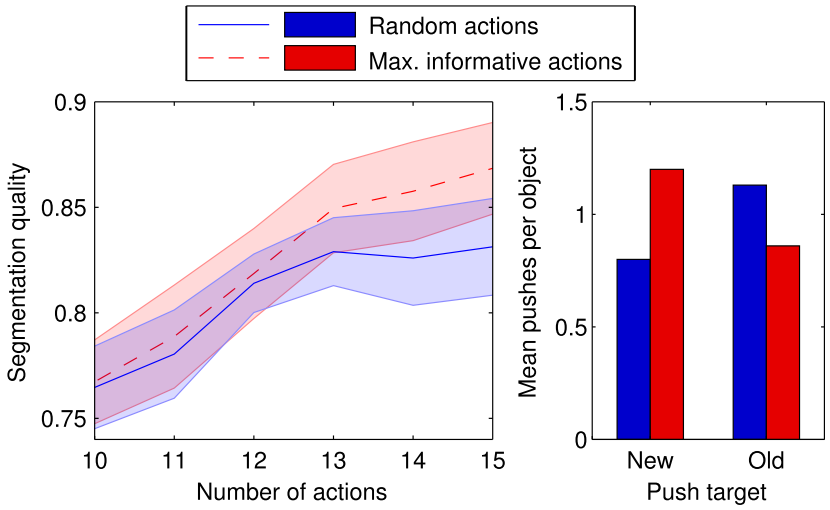


Figure 2.13.: Action selection experiment. On the left, the segmentation quality is shown starting from the tenth action, when two objects are moved inside the robot's workspace. The right graph shows the mean number of pushes directed at existing and newly introduced objects during this period. The shaded areas show the standard error.

those objects tended to be closer to the robot after the initial pushes, so more of their parts were reachable to be pushed. The action selection strategy employing the mutual information criterion focused explorative actions on the novel objects, overcoming the bias observed in the random strategy.

Analysis of Segmentation Errors. Our probabilistic interactive segmentation approach, whether employing actions deemed maximally informative or not, often does not attain the maximally possible segmentation quality of 1.0. We investigated a number of possible causes to understand where our setup or algorithms can be improved.

First of all, we looked at our sampling method. Multi-modal posterior distributions can cause Markov Chain samplers to get stuck in suboptimal local maxima of the likelihood function. In that case, we expect chains with different starting points to reach different local maxima. We compared chains starting with all parts in their own cluster, all parts in a single cluster, or with the ground truth (ensuring we are near a good local maximum). The mean segmentation quality after observing 15 actions was, respectively, 0.79, 0.80, and 0.80, with a standard deviation of 0.07 in each case. The likelihood values of the samples were comparable as well. Therefore, local maxima are unlikely to explain failures in our experiments.

Next, we looked at tracking errors. Sometimes, the tracker seems to recognize a part but returns the wrong location. Such false positives only occur in about 1% of the parts during each trial run, and can be handled as noise by our method. Losing track of parts happen more often, e.g. during occlusions or illumination changes (Tab. 2.1). Occasionally failures can be handled by our method, but about 11% of the parts are lost for more than half of the 15 time steps, which decreases the system's performance. Tracker performance is therefore a target for future improvement (Sec. 2.3.2).

Time steps lost, #	0	1-3	4-7	8-14
Occurrence, %	65%	15%	9%	11%
Mean error contribution	-1.6	-2.1	-2.3	-4.8

Table 2.1.: The tracker sometimes fails to localize parts. The contribution is the number of pairs of parts including the lost part that do not contribute to $|P \cap Q|$ in the quality measure (Equation 2.4). Shown are only parts that moved at least once during the experiment to avoid confounding movement and tracking success.

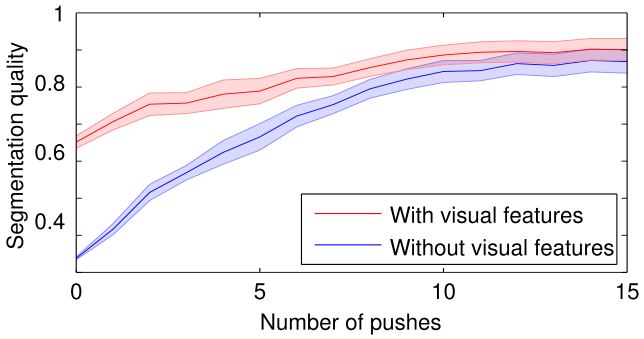


Figure 2.14.: Experiment with random actions and setups with a cluster of four objects. Especially when little interaction data is available, knowledge about visual features (spatial proximity) helps to infer the correct segmentation. Shaded areas show the standard error.

2.4.4 Integration of Motion and Visual Clues Experiment

As suggested in Sec. 2.3.4, we combined the observed movement data with visually observed properties, in this case the spatial arrangements of objects in the beginning of the experiment. The parameters of the likelihood model are set by maximizing the marginal likelihood on a separate data set of 20 scenes. The likelihood parameters of the motion model were again set to their MAP values. We performed two experiments with different action selection criteria.

Random Actions. We evaluated the model with and without the additional visual clues on 10 sequences from the data set used in Sec. 2.4.2 for which the visual information was available (in each, 15 random actions were performed). The results of this experiment are shown in Figure 2.14.

Informative Actions. We performed twelve trials with the objects set up in two clusters: one containing three objects and the other with a single object (from the set in Figure 2.8). In this setup, we evaluate how action selection using the mutual information criterion can exploit information from static features. We hypothesize that exploration will focus on the larger cluster which is visually more ambiguous. Results from this experiment are shown in Figures 2.15 and 2.16.

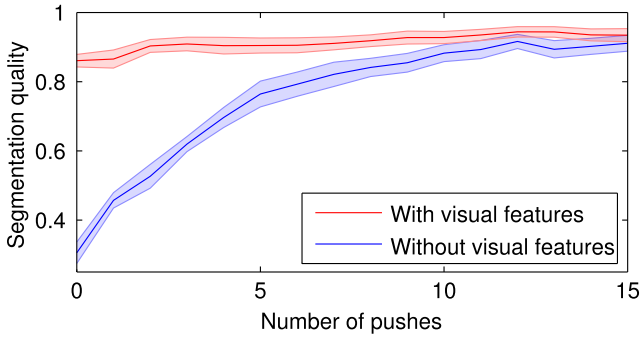


Figure 2.15.: Experiment with mutual information action selection and setups with one cluster of three objects and one isolated object. On scenes where one object is initially separated, visual features are more powerful. Shaded areas show the standard error.

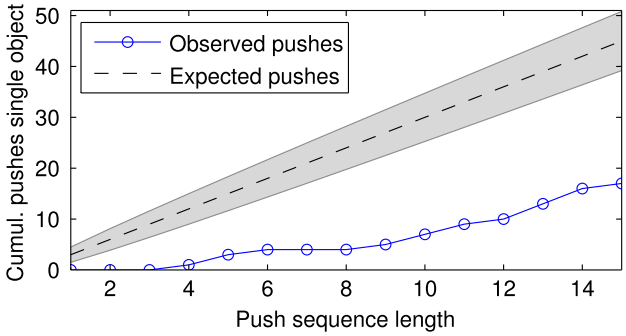


Figure 2.16.: Experiment with mutual information action selection and setups with one cluster of three objects and one isolated object. We show the cumulative count of pushes targeted at the isolated object during the indicated sequence length. The shaded area represents the standard deviation of a binomial distribution.

Discussion. At the start of the experiments, no interaction data were available so the baseline approach guessed blindly. When using the static visual features, we obtained much better results. As interaction data became available, the gap between the approaches diminished but the method that employed additional static visual features stayed numerically better.

As shown in Figures 2.15 and 2.16, in case we started with one object separated from the other objects, the robot avoided this object in subsequent exploration. Only after exploring the remaining cluster of objects, the robot targeted the single object more frequently.

In related work, visual clues were often used to provide a set of hypothesis to be confirmed by interaction [Bergström et al., 2011, Schiebener et al., 2013, Hausman et al., 2013]. In contrast, in our method we regard both visual and interactive clues as (noisy) information channels. Instead of maintaining a discrete set of hypothesis, our method assigns a, usually non-zero, probability to any possible segmentation. This view allows information fusion from both channels in a principled way. Parameters for the visual feature model were determined by training on previous (known) scene segmentations, allowing knowledge to be transferred between scenes. Hence,

no manual tuning was needed for setting the hyper-parameters of the visual feature model and for the integration of visual features and interaction features.

2.5 Conclusion

To conclude this chapter, we will give a short summary of proposed method and the experimental findings. Then, the epilogue will cover suggestions for future work, and an overview of work in the field performed since this chapter was first published.

2.5.1 Summary of this Chapter

In this chapter, we have presented a part-based, probabilistic approach to interactive segmentation. Our approach aims at minimizing human intervention: the robot learns from the effects of its actions, rather than human-given labels, and the amount of tuning needed is limited by employing Bayesian methods (enabling important parameters to be marginalized given largely uninformative hyper-priors) and machine learning approaches to parameter setting (maximum marginal likelihood).

Our experiments showed that, firstly, our approach functions and is relatively robust to noise and co-movement, even in a complicated real-world environment including tracking failures and co-movement of different objects. As we employ a probabilistic representation, our robot has knowledge on the segmentation uncertainty. This knowledge can be exploited to select more informative actions. In a second experiment, we have shown that our information-theoretic scheme for action selection enables the robot to learn faster about new objects in its environment then a random baseline, by directing more explorative actions at those novel objects.

Another advantage of a probabilistic representation is that it offers a straightforward way to integrate clues for different sources, as conditionally independent clues can be integrated by multiplying their likelihood functions together. Specifically, we studied a spatial proximity feature. We avoided hand-tuning of the hyper-parameters of the spatial likelihood model by learning them from a set of scenes the robot has previously interacted with. The learned parameters allowed knowledge on typical spatial structures to be transferred to new scenes. We have shown that this transfer makes determination of the underlying segmentation substantially faster and can focus action selection to the most ambiguous parts of the scene.

2.5.2 Epilogue

This chapter addressed the problem of quantifying uncertainty in interactive segmentation, and choosing maximally informative actions. Sections 2.1–2.5.1 of this chapter have previously been published in the IEEE Transactions on Robotics [van Hoof et al., 2014]. That publication in itself is based on earlier conference papers [van Hoof et al., 2012, 2013]. A lot of work in the area of interactive segmentation has been published since the work in this chapter was originally published. A recent overview of interactive perception methods is given in the review by Bohg et al. [2016]. The discussion in this section will be focused on the two most relevant bodies of work.

First, we will discuss some methods that extend and refine capabilities in interactive segmentation. Then, we will discuss recent work that analyzes different objectives for selecting informative actions, and work in interactive perception that employs such methods.

In interactive segmentation, work has extended the state of the art in multiple directions. Schiebener et al. [2014], who use a combination of computer vision methods to generate hypotheses and interaction to validate those hypotheses, introduced an improved tracker that is able to handle textureless object candidates. On the other hand, Xu et al. [2015] combine segmentation with object modeling, whilst selecting actions according to the information gain of both the segmentation and the reconstruction jointly.

The robot's final goal is usually not to obtain a segmentation. Pajarinen and Kyrki [2015] introduce a method that optimally trades off exploring the underlying segmentation and reaching a manipulation goal. After segmentation, the next step in an interactive perception process could be, to determine how an object can be manipulated, e.g. by identifying its articulate structure [Martín-Martín and Brock, 2014, Hausman et al., 2015, Höfer and Brock, 2016, Otte et al., 2014, Kulick et al., 2015].

Since our work was published, multiple researchers have used comparable information gain criteria for selecting informative actions. For segmentation and modeling, Xu et al. [2015] minimized posterior joint entropy over segmentations and reconstructions. To learn affordances, Otte et al. [2014] used a Markov chain Monte Carlo approach to likewise estimate the posterior entropy, whereas Hausman et al. [2015] compared a posterior entropy criterion to the following criterion based on the Kullback Leibler divergence $D_{\mathrm{KL}}(p(\mathbf{s}|\mathcal{D})||\mathbb{E}_{\mathbf{o}}[p(\mathbf{s}|\mathcal{D},\mathbf{o},\mathbf{a})|\mathbf{a}])$. Kulick et al. [2015, 2014] proposed using the cross-entropy $\mathbb{E}_{\mathbf{o}}[H(p(\mathbf{s}|\mathcal{D});p(\mathbf{s}|\mathcal{D},\mathbf{o},\mathbf{a})|\mathbf{a}])$. The work of Kim and Sukhatme [2015] also employs an information gain-based objective, but is somewhat vague in the exact definition.

Furthermore, a recent theoretical analysis has shown, that maximizing the expected Kullback-Leibler divergence between current and future models minimizes the difference between the future model and the true state of the world [Little and Sommer, 2013, Mobin et al., 2014]. Although that work focuses on the more complicated MDP setting, their main argument is that the quantity to be maximized is the decrease in divergence from learned model distribution $p(s|\mathcal{D})$ to the true model distribution, which is, in this case, is a Dirac delta $p_{\mathrm{true}}(\mathbf{s}) = \delta(\mathbf{s}-\mathbf{s}^*)$:

$$\begin{aligned}
&\arg\max_{\mathbf{a}} \mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{true}}(\mathbf{s})||p(\mathbf{s}|\mathcal{D})\right) - \mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{true}}(\mathbf{s})||p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})\right) \\
&= \arg\max_{\mathbf{a}} \mathbb{E}_{\mathbf{s}\sim p_{\mathrm{true}}} \log\left(\frac{p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})}{p(\mathbf{s}|\mathcal{D})}\right) = \arg\max_{\mathbf{a}} \log\left(\frac{p(\mathbf{s}^*|\mathcal{D},\mathbf{a},\mathbf{o})}{p(\mathbf{s}^*|\mathcal{D})}\right)
\end{aligned}$$

Since neither $\mathbf{s}^*$ or the observation $\mathbf{o}$ resulting from action $\mathbf{a}$ are known, the agent can only evaluate a Bayesian estimate of this objective, based on its current knowledge, the predicted information gain:

$$\mathbb{E}_{\mathbf{s},\mathbf{o}}\left[\log\left(\frac{p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})}{p(\mathbf{s}|\mathcal{D})}\right)\middle|\,\mathcal{D},\mathbf{a}\right] = \mathbb{E}_{\mathbf{o}}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})||p(\mathbf{s}|\mathcal{D}))|\mathcal{D},\mathbf{a}\right].$$

This result corresponds to the selection criteria that we proposed in this chapter. Since the information gain is adaptive submodular, greedy policies are guaranteed to obtain near-optimal performance [Vien and Toussaint, 2015]. Furthermore, since

$$\mathbb{E}_{\mathbf{o}}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})||p(\mathbf{s}|\mathcal{D}))|\mathcal{D},\mathbf{a}\right] = H(\mathbf{s}|\mathcal{D}) - \mathbb{E}_{\mathbf{o}}[H(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o})|\mathbf{a}],$$

where H denotes the differential entropy or conditional entropy, maximizing the proposed Kullback-Leibler divergence is *equivalent* to minimizing the expected posterior entropy (since $H(p(\mathbf{s}|\mathcal{D}))$ is independent of the chosen action). Thus, the criterion used in this chapter yields the same actions as the expected conditional entropy used in other approaches [Xu et al., 2015, Otte et al., 2014, Hausman et al., 2015].

On the other hand, Kulick et al. [2015, 2014] propose a cross-entropy method that, they show, is equivalent to maximizing the KL-divergence from the posterior to the prior

$$\arg\max_{\mathbf{a}} \mathbb{E}_{\mathbf{o}}[H\left(p(\mathbf{s}|\mathcal{D}); p(\mathbf{s}|\mathcal{D},\mathbf{o},\mathbf{a})\right)|\mathbf{a}] = \mathbb{E}_{\mathbf{o}}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{s}|\mathcal{D})||p(\mathbf{s}|\mathcal{D},\mathbf{a},\mathbf{o}))|\mathcal{D},\mathbf{a}\right].$$

On the basis of example, they argue that sequential greedy optimization of this objective is less likely to get stuck in global minima when the prior distribution is misleading. The criterion proposed by Hausman et al. [2015] based on the Kullback-Leibler divergence $D_{\mathrm{KL}}(p(\mathbf{s}|\mathcal{D})||\mathbb{E}_{\mathbf{o}}[p(\mathbf{s}|\mathcal{D},\mathbf{o},\mathbf{a})|\mathbf{a}])$ from posterior to prior, yielded better results in that study. Since the expectation appears inside the divergence, this criterion is not directly related to the expected information gain or cross-entropy criterion. One possible reason for the relatively good performance obtained using this criterion compared to the posterior entropy in that study, is that a potentially fragile density estimation step was used to evaluate the posterior entropy. The possibility of information gain optimization to get stuck in local optima is another possible reason [Kulick et al., 2015, 2014].

Possible future research topics to improve performance include improving tracking performance, integrating more visual clues, such as color clues, and making the Markov chain more efficient by a larger variety of moves.

More generally, learning to segment and reconstruct objects together with learning how to reach a manipulation goal would allow robots to learn how to perform simple domestic task autonomously. Reaching this challenging goal would require using more flexible probabilistic models, such that a larger variety of actions and consequences can be modeled.

To allow the methods to scale up, more prior knowledge should be integrated, for example about the physics of pushing. Another possible area for future work is to calculate the long-term rather than the one-step expected information gain. To do this efficiently, (approximate) dynamic programming in the belief space could be considered.

3 Non-parametric Policy Search with Limited Information Loss

After considering an exploration task without an explicit goal in the last chapter, in this chapter we will consider learning goal-directed behavior. Like in the previous chapter, the robot again observes the world and chooses actions. However, in addition the robot will receive a weak supervision signal consisting of a single real scalar indicating a cost or reward specifying the task's goal. The robot aims to maximize the obtained rewards (respectively, minimize costs).

Learning control policies from such rewards is formalized in the Markov decision process (MDP) framework, and many solutions have been proposed by, e.g., the reinforcement learning community. However, typical solutions focus on discrete MDPs with rather few states. Real robots get continuous and possibly high-dimensional input from there sensor data. This data yields a couple of problems.

First of all, representing relevant objects, such as value functions (that represent the desirability of a state) or control policies, becomes hard. Sometimes, a low-dimensional designed feature basis is available [Kaelbling et al., 1996, Kober et al., 2013, Bartlett, 2003]. However, in many situations this is not the case. Instead of a low-dimensional designed features, generic features such as a polynomial or radial basis functions (RBFs) could be used. However, such features yield impractically many features for input with a high dimensionality.

This yields a second, related problem. When the number of samples (data points) is low compared to the number of basis functions, approximations of, e.g., value functions can deviate quite far from their true value. These deviations are typically larger in areas of the state space where few or no samples are present. Trying to select the best actions by optimizing an approximated value function can thus yield actions far from the real optimum.

In policy search algorithms, the sampling policy is explicitly represented, so that the updated policy can be constrained to be close to the previous policy. This approach forces the policy to stay close to previous samples, where the approximation of, e.g., value functions tends to be more reliable. In this chapter, we extend such an algorithm to use non-parametric value functions, transition models, and control policies. Non-parametric representations can implicitly use infinite-dimensional basis features, of which only those basis features that coincide with data points will get non-zero weights. As such, by design, such methods ignore parts of the huge sensory space where no data points are observed, and focus the representative power on parts of the state space where data points are densely concentrated, and are thus are most relevant for the task at hand.

3.1 Introduction

Learning continuous valued control policies directly from sensory input presents a major obstacle to applying reinforcement learning (RL) methods effectively in realistic settings. In such settings, there exist two major problems. First, sensory inputs are often high-dimensional, making discretization of the state-space infeasible. Secondly, in such settings, the amount of data that can be used to learn policies is often severely limited, increasing the bias in policy updates which can lead to oscillations or divergence [Mannor et al., 2007, Peters et al., 2008]. For the first problem, algorithms have been developed that rely on human-designed features for value function approximations or specialized parametric policies [Kaelbling et al., 1996, Kober et al., 2013, Bartlett, 2003]. However, the dependence on human engineering limits the applicability of such methods, especially in high-dimensional sensory domains where defining good features is non-trivial.

Recently, there has been a lot of progress towards avoiding the dependence on engineered features by using non-parametric techniques. Such techniques often use kernel functions to implicitly define a (possibly infinite) features space, replacing the manual definition of features. In contrast to task-specific hand-tuned feature spaces, many popular kernels are applicable to a large number of problems as the resulting representation can adapt to the complexity of the data. Non-parametric techniques have been successfully used in value-function methods, for example by Grünewälder et al. [2012b], Nishiyama et al. [2012], and Kroemer and Peters [2011]. An overview of related work on non-parametric methods is given in Section 3.4.3 on page 69. One shortcoming of such methods is that they generally require inverting matrices that grow with the number of data points, which limits their applicability. Another shortcoming is that these methods are still susceptible to the second problem of premature convergence due to data scarcity.

The problem of data scarcity is aggravated by the lack of a notion of the sampled data or sampling policy in most reinforcement learning approaches. Instead, most methods directly optimize the expected return. Thus, policy updates frequently do not have a mechanism to stay close to the observed data. Optimization of the policy with respect to the expected reward automatically means the improved policy needs *to forget experience* to avoid the mistakes of the past and to replicate the observed successes. Consequentially, policy updates that do not stay close to the observed data will often result in a loss of essential information. For example, a policy update that eliminates most exploration by taking the best observed action often yields fast but premature convergence to a suboptimal policy [Kakade, 2002].

Moreover, choosing an improved policy purely based on sampled returns favors biased solutions that eliminate states in which only bad actions have been tried out. This problem is known as *optimization bias* [Mannor et al., 2007]. Optimization biases may appear in both on- and off-policy reinforcement learning methods due to under-sampling (e.g., if we cannot sample sufficiently many of the state-actions pairs prescribed by a policy, we will over-fit), model errors or even the policy update step itself.

In an on-line setting, many methods address this problem implicitly by staying close to the previous policy. For example, policy gradient methods allow only small incremental policy changes. The Fisher information metric—that occurs in policy updates using the natural policy gradients [Kakade, 2002, Peters and Schaal, 2008b]—can be seen as a Taylor expansion of the loss of information or *relative entropy* between the path distributions generated by the original and the updated policy [Bagnell and Schneider, 2003a]. Instead of bounding this Taylor approximation, we can explicitly bound the relative entropy between successive state-action distributions, leading to the Relative Entropy Policy Search (REPS) algorithm. We discuss policy search methods that limit the information loss in Section 3.4.1 on page 65.

In this paper, we propose a method based on this insight, that allows us to compute new policies given a data distribution both for off-policy or on-policy reinforcement learning. We start from the optimal control problem statement subject to the constraint that the loss in information is bounded. For continuous domains, where a suitable set of features is often not available, we develop a non-parametric version of this algorithm. This algorithm uses general kernels to define (possibly infinite) feature spaces implicitly, and consider ways to efficiently approximate this method to make it applicable to large datasets.

In our experiments, we show that our method outperforms relevant baselines on a reaching task and an underpowered swing-up task. Furthermore, we evaluate different approximations to process larger data sets efficiently. Finally, we show that using such an approximation, a real-robot pendulum swing-up task can be learned from high-dimensional vision data.

3.1.1 Problem Statement and Notation

In a Markov decision process (MDP), an agent in state $\mathbf{s}$ selects an action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ according to a (possibly stochastic) policy π and receives a reward $\mathcal{R}_{\mathbf{s}}^{\mathbf{a}} \in \mathbb{R}$. We will assume continuous state-action spaces: $\mathbf{s} \in \mathcal{S} = \mathbb{R}^{D_\mathrm{s}}$, $\mathbf{a} \in \mathcal{A} = \mathbb{R}^{D_\mathrm{a}}$. If the Markov decision process is ergodic, for each policy π, there exists a stationary distribution $\mu_\pi(\mathbf{s})$ such that $\int_\mathcal{S}\int_\mathcal{A} \mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})\mu_\pi(s)\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} = \mu_\pi(\mathbf{s}')$, where $\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}} = p(\mathbf{s}'|\mathbf{a},\mathbf{s})$. The goal of a reinforcement learning agent is to choose a policy such that the joint state-action distribution $p_\pi(\mathbf{s},\mathbf{a}) = \mu_\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$ maximizes the average reward $J(\pi) = \int_\mathcal{S}\int_\mathcal{A}\ \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})\mathcal{R}_{\mathbf{s}}^{\mathbf{a}}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}$.

The goal of relative entropy policy search is to obtain policies that maximize the expected reward $J(\pi)$ while bounding the information loss, i.e.,

$$\max_{\pi,\mu_\pi} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})\mathcal{R}_{\mathbf{s}}^{\mathbf{a}}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}, \tag{3.1}$$

$$\text{s. t.} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} \quad = 1, \tag{3.2}$$

$$\forall s'. \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} = \mu_\pi(\mathbf{s}'), \tag{3.3}$$

$$\mathrm{D}_\mathrm{KL}(\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})||q(\mathbf{s},\mathbf{a})) \ \leq \epsilon, \tag{3.4}$$

where Eqs. (3.1-3.3) specify the general reinforcement learning objective (3.1) with the constraints that $\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ is a distribution (Eq. 3.2) and μ_π is the stationary distribution under policy π (Eq. 3.3). Equation (3.4) specifies the additional bound on the KL divergence, where

$$\mathrm{D}_{\mathrm{KL}}(p(x)||q(x)) = \int p(x)\log(p(x)/q(x))\mathrm{d}x.$$

The reference distribution q is usually set to the state-action distribution induced by previous policies, where the initial explorative policy is a wide, uninformed distribution such as a zero-centered Gaussian with a larger variance. In each iteration, the policy is adapted to maximize the expected reward while respecting the constraint on the KL divergence. Thus, as learning progresses, the policy typically slowly converges towards a deterministic reward-maximizing policy.

In this chapter, we aim at developing a reinforcement learning algorithm applicable in continuous state-action MDPs with high-dimensional state representations. We assume hand-coded feature functions and parametric policies are not available and the transition and reward models of the MDP are unknown. Furthermore, we will concentrate on infinite-horizon problems.

3.2 Stable Policy Updates for Stochastic Continuous MDPs

Solutions to the relative entropy policy search optimization problem in Equations (3.1)-(3.4) provide stable controller updates. However, for continuous systems with stochastic dynamics and non-parametric controllers, it is not straightforward to optimize the optimization problem directly. In this section, we explain the steps to obtain a practical algorithm. First, we show how to find the dual of the optimization problem. Subsequently, we discuss how this problem can be solved for stochastic systems that are continuous and non-linear. After that, we discuss how to relax the assumption of ergodicity of the MDP by transforming the average reward MDP in a discounted reward MDP. Solving the optimization problem results in a new optimal policy that is, however, only defined on the current set of samples. Therefore, we discuss how the sample-based optimal policy can be generalized to the entire state space. Finally, we will discuss how to set the hyper-parameters of the different steps of our method with minimal manual tuning.

3.2.1 Finding the Dual Problem

To find the dual to the optimization problem in Equations (3.1)–(3.4), first, we formulate the Lagrangian. For every constraint, we introduce a Lagrangian multiplier. Because Eq. (3.3) represents a continuum of constraints, we introduce a corresponding continuous state-dependent Lagrangian multiplier $V(\mathbf{s})$. Instead of summing the contributions for each constraints in the dual function, here, we have to integrate in-

stead. We will write $p_\pi(\mathbf{s},\mathbf{a}) = \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ to keep the exposition brief. Therefore, the Lagrangian

$$L(p,\eta,V,\lambda) = \iint_{\mathcal{A}\times\mathcal{S}} p_\pi(\mathbf{s},\mathbf{a})\mathcal{R}_\mathbf{s}^\mathbf{a}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} + \int_\mathcal{S} V(\mathbf{s}')\left(\iint_{\mathcal{A}\times\mathcal{S}} p_\pi(\mathbf{s},\mathbf{a})\mathcal{P}_{\mathbf{s}\mathbf{s}'}^\mathbf{a}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} - \mu_\pi(\mathbf{s}')\right)\mathrm{d}\mathbf{s}' + \lambda\left(1 - \iint_{\mathcal{A}\times\mathcal{S}} p_\pi(\mathbf{s},\mathbf{a})\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}\right) + \eta\left(\epsilon - \iint_{\mathcal{A}\times\mathcal{S}} p_\pi(\mathbf{s},\mathbf{a})\log\frac{p_\pi(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}\right).$$

Using the identity $\mu_\pi(\mathbf{s}) = \int_A p_\pi(\mathbf{s},\mathbf{a})\mathrm{d}\mathbf{a}$, the Lagrangian can be re-shaped in the more convenient form

$$L(p,\eta,V,\lambda) = \lambda - \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}[V(\mathbf{s})] + \eta\epsilon + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}\left[\mathcal{R}_\mathbf{s}^\mathbf{a} - \lambda + \int_\mathcal{S} V(\mathbf{s}')\mathcal{P}_{\mathbf{s}\mathbf{s}'}^\mathbf{a}\mathrm{d}\mathbf{s}' - \eta\log\frac{p_\pi(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})}\right].$$

To find the optimal p, we take the derivative of L w.r.t. p and set it to zero

$$\frac{\partial L}{\partial p_\pi(\mathbf{s},\mathbf{a})} = \mathcal{R}_\mathbf{s}^\mathbf{a} - \lambda + \int_\mathcal{S} V(\mathbf{s}')\mathcal{P}_{\mathbf{s}\mathbf{s}'}^\mathbf{a}\mathrm{d}\mathbf{s}' - \eta\log\frac{p_\pi(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})} - \eta - V(\mathbf{s}) = 0.$$

Therefore, we obtain the new state-action distribution

$$p_\pi(\mathbf{s},\mathbf{a}) = q(\mathbf{s},\mathbf{a})\exp\left(\frac{\mathcal{R}_\mathbf{s}^\mathbf{a} + \int_\mathcal{S} V(\mathbf{s}')\mathcal{P}_{\mathbf{s}\mathbf{s}'}^\mathbf{a}\mathrm{d}\mathbf{s}' - V(\mathbf{s})}{\eta}\right)\exp\left(\frac{-\lambda-\eta}{\eta}\right). \tag{3.5}$$

The function $V(\mathbf{s})$ resembles a value function, such that $\delta(\mathbf{s},\mathbf{a},V) = \mathcal{R}_\mathbf{s}^\mathbf{a} + \int_\mathcal{S} V(\mathbf{s}')\mathcal{P}_{\mathbf{s}\mathbf{s}'}^\mathbf{a}\mathrm{d}\mathbf{s}' - V(\mathbf{s})$ can be identified as a Bellman error. Since $p_\pi(\mathbf{s},\mathbf{a})$ is a probability distribution, we can identify $\exp(-\lambda/\eta - 1)$ to be a normalization factor

$$\exp\left(-\frac{\lambda}{\eta} - 1\right) = \frac{1}{\iint_{\mathcal{A}\times\mathcal{S}} q(\mathbf{s},\mathbf{a})\exp(\delta(\mathbf{s},\mathbf{a},V)/\eta)\,\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}} = \frac{1}{\mathbb{E}_{q(\mathbf{s},\mathbf{a})}\exp(\delta(\mathbf{s},\mathbf{a},V)/\eta)},$$

which yields the policy

$$\pi(\mathbf{a}|\mathbf{s}) \propto q(\mathbf{s},\mathbf{a})\exp\left(\frac{\delta(\mathbf{s},\mathbf{a},V)}{\eta}\right). \tag{3.6}$$

To obtain the dual function, we re-insert the state-action probabilities $p_\pi(\mathbf{s},\mathbf{a})$ in the Lagrangian

$$\begin{aligned} g(\eta,V,\lambda) =& \lambda + \eta\epsilon + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}\left[\delta(\mathbf{s},\mathbf{a},V) - \lambda - \eta\log\frac{p_\pi(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})}\right] \\ =& \eta\epsilon + \eta\log\left(\mathbb{E}_{q(\mathbf{s},\mathbf{a})}\exp(\delta(\mathbf{s},\mathbf{a},V)/\eta)\right), \end{aligned}$$

We typically do not know the sampling distribution q, as it depends on the unknown system dynamics. However, the expected value over q can be approximated straightforwardly by taking the average of samples $1,\ldots,n$ taken from q. Note that λ and q do not appear in the final expression of the dual function

$$g(\eta,V) = \eta\epsilon + \eta\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp(\delta(\mathbf{s}_i,\mathbf{a}_i,V)/\eta)\right). \tag{3.7}$$

To compute the Bellman error δ, the transition distribution is required. As this distribution is generally not known, δ needs to be approximated. The dual function (3.7) depends implicitly on reference distribution q through the samples.

3.2.2 Solving the Dual Problem

To solve Eqs. (3.1)–(3.4), we need to find the Lagrangian parameters that minimize the dual function in Equation (3.7), i.e., $(\eta^*, V^*) = \arg\min g(\eta, V)$. V is a function with domain $\mathcal{S}$, hence, for continuous domains, we will surely over-fit without additional assumptions. One possibility would be to assume V a function linear in designed features, but good features are task-specific and often hard to define. We will therefore make the more general assumption that V^* is of the form

$$V^*(\cdot) = \sum_{\tilde{\mathbf{s}} \in \tilde{\mathcal{S}}} \alpha_{\tilde{\mathbf{s}}} k_s(\tilde{\mathbf{s}}, \cdot), \tag{3.8}$$

for any set of states $\tilde{\mathcal{S}}$ and scalars α, and a chosen reproducing kernel k_s. In other words, we assume $V^* \in \mathcal{F}$ for a reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ with kernel k_s. The kernel k_s implicitly defines a (possibly infinite dimensional) feature map $\boldsymbol{\phi}(\mathbf{s}) = k_s(\mathbf{s}, \cdot)$ [Schölkopf et al., 1999]. Such an implicit definition has the advantage that we do not need to explicitly compute a feature basis for V^* [Hofmann et al., 2008]. Kernels are in most cases easier to choose than feature vectors as the complexity of V^* can grow with the amount of training data. In the final algorithm, we will only work with inner product of implicit features that can be computed using the kernel function, $\boldsymbol{\phi}(\mathbf{s})^T \boldsymbol{\phi}(\mathbf{s}') = k_s(\mathbf{s}, \mathbf{s}')$. Therefore, we do not need to explicitly represent the (possibly infinite dimensional) feature maps [Hofmann et al., 2008, Schölkopf et al., 1999].

Embedding the Transition Model. Since the transition model $\mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'}$ in Eq. (3.5) is unknown, we need to approximate $\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s}, \mathbf{a}]$. To do so, we embed the conditional distribution $\mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'}$ in the RKHS $\mathcal{F}$, i.e., we represent $\mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'}$ by the expected implicit features $\boldsymbol{\mu}_{\mathbf{s}'|\mathbf{s},\mathbf{a}} = \mathbb{E}_{\mathbf{s}'}[\boldsymbol{\phi}(\mathbf{s}')|\mathbf{s}, \mathbf{a}]$. Using such embeddings avoids estimating the joint density and leads to good results even for high-dimensional data [Song et al., 2013]. They also render calculations of expected values over a function in $\mathcal{F}$ straightforward without numerical integration [Song et al., 2013]. In order to learn the conditional operator, we will use a kernel over the state-action space of the form $\boldsymbol{\psi}(\mathbf{s}, \mathbf{a}) = k_s(\mathbf{s}, \cdot) k_a(\mathbf{a}, \cdot)$. Given a sample $\{(\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}'_1), \ldots, (\mathbf{s}_n, \mathbf{a}_n, \mathbf{s}'_n)\}$, the empirical conditional embedding is defined as

$$\hat{\boldsymbol{\mu}}_{S'|s,a} = \hat{\mathbf{C}}_{S'|S,A} \boldsymbol{\psi}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\Phi} \boldsymbol{\beta}(\mathbf{s}, \mathbf{a}), \tag{3.9}$$

$$\hat{\mathbf{C}}_{S'|S,A} = \boldsymbol{\Phi}(\mathbf{K}_{\text{sa}} + \lambda \mathbf{I})^{-1} \boldsymbol{\Psi}^T, \tag{3.10}$$

where $\hat{\mathbf{C}}_{S'|S,A}$ is a learned conditional operator that allows the computation of embedding strengths $\boldsymbol{\beta}(\mathbf{s}, \mathbf{a}) = (\mathbf{K}_{\text{sa}} + \lambda \mathbf{I})^{-1} \mathbf{k}_{\text{sa}}(\mathbf{s}, \mathbf{a})$, as suggested by Grünewälder et al. [2012a,b]. In this equation, λ is a regularization coefficient, the matrices $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\mathbf{s}_1, \mathbf{a}_1), \ldots, \boldsymbol{\psi}(\mathbf{s}_n, \mathbf{a}_n)]$ and $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{s}'_1), \ldots, \boldsymbol{\phi}(\mathbf{s}'_n)]$ consist of implicit feature factors, whereas matrix $\mathbf{K}_{\text{sa}} = \boldsymbol{\Psi}^T \boldsymbol{\Psi}$ and vector $\mathbf{k}_{\text{sa}}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\Psi}^T \boldsymbol{\psi}(\mathbf{s}, \mathbf{a})$ contain kernel function evaluations between pairs of data points.[1]

[1] This means $[\mathbf{K}_{\text{sa}}]_{ij} = k_s(\mathbf{s}_i, \mathbf{s}_j) k_a(\mathbf{a}_i, \mathbf{a}_j)$, $[\mathbf{k}_{\text{sa}}(\mathbf{s}, \mathbf{a})]_i = k_s(\mathbf{s}_i, \mathbf{s}) k_a(\mathbf{a}_i, \mathbf{a})$.

Functional Form of V. The function k_s is a reproducing kernel and $V \in \mathcal{F}$, i.e., is of the form (3.8). Therefore, the expected value of V can be approximated using the embedded distribution [Song et al., 2013, Grünewälder et al., 2012a,b], i.e.,

$$\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}] = \left\langle V, \hat{\mu}_{S'|s,a} \right\rangle_{\mathcal{F}} = \sum_{i=1}^{n} \beta_i(\mathbf{s},\mathbf{a}) V(\mathbf{s}'_i).$$

In the dual function g from Equation (3.7), V is now only evaluated at sampled states $\mathbf{s}_i$ and $\mathbf{s}'_i$. As we assumed $V \in \mathcal{F}$, the generalized representer theorem [Schölkopf et al., 2001] tells us that there is at least one optimum of the form (3.8) with $\tilde{\mathcal{S}}$ the set of all sampled states[2]. Consequently, $\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}] - V(\mathbf{s}) = \boldsymbol{\alpha}^T \tilde{\mathbf{K}}_s \boldsymbol{\beta}(\mathbf{s},\mathbf{a}) - \boldsymbol{\alpha}^T \mathbf{k}_{s(\mathbf{s})}$, where $\tilde{\mathbf{K}}_s$ is the Gram matrix with entries $[\tilde{\mathbf{K}}_s]_{ji} = k_s(\tilde{\mathbf{s}}_j, \mathbf{s}'_i)$, and $[\mathbf{k}_s(\mathbf{s})]_j = k_s(\tilde{\mathbf{s}}_j, \mathbf{s})$.

Finding a Numerical Solution. The dual problem can now be restated in terms of η and $\boldsymbol{\alpha}$, as

$$g(\eta,\boldsymbol{\alpha}) = \eta\epsilon + \eta \log\left(\sum_{i=1}^{n} \frac{1}{n} \exp\left(\frac{\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})}{\eta}\right)\right), \tag{3.11}$$

$$\delta(\mathbf{s},\mathbf{a},\boldsymbol{\alpha}) = \mathcal{R}^{\mathbf{a}}_{\mathbf{s}} + \boldsymbol{\alpha}^T \left(\tilde{\mathbf{K}}_s \boldsymbol{\beta}(\mathbf{s},\mathbf{a}) - \mathbf{k}_s(\mathbf{s})\right). \tag{3.12}$$

This objective is convex in $\boldsymbol{\alpha}$. Since the analytic gradient and Hessian for this objective are straightforward to obtain[3], we employ second order optimization methods to find the optimal η and $\boldsymbol{\alpha}$. We employ an iterative optimization scheme similar to the one described by Lioutikov et al. [2014], that is, we sequentially optimize for either η or $\boldsymbol{\alpha}$ until the constraints are fulfilled within an acceptable tolerance.

If we choose kernels $\mathbf{k}_s(\mathbf{s}_i,\mathbf{s}_j) = \tilde{\phi}(\mathbf{s}_i)^T \tilde{\phi}(\mathbf{s}_j)$ and $\mathbf{k}_{sa}((\mathbf{s}_i,\mathbf{a}_i),(\mathbf{s}_j,\mathbf{a}_j)) = \mathbb{1}_{\{(\mathbf{s}_i,\mathbf{a}_i)\}}((\mathbf{s}_j,\mathbf{a}_j))$, we obtain the original REPS formulation [Peters et al., 2010] as a special case, with corresponding Bellman error

$$\delta(\mathbf{s},\mathbf{a},\boldsymbol{\alpha}) = \mathcal{R}^{\mathbf{a}}_{\mathbf{s}} + \boldsymbol{\alpha}^T(\tilde{\phi}(\mathbf{s}') - \tilde{\phi}(\mathbf{s})). \tag{3.13}$$

In these equations, $\mathbb{1}$ is the indicator function, $\tilde{\phi}$ is a set of hand-crafted features, and $\mathbf{s}'$ is the observed outcome of applying action $\mathbf{a}$ in state $\mathbf{s}$. Our generalization allows the selection of widely applicable kernels that do not depend on hand-crafted features. Furthermore, avoiding the identity kernel function over the state-action space allows efficient learning in stochastic systems.

3.2.3 Ensuring a Stationary Distribution

The REPS formulation (Eqs. 3.1–3.4) assumes the existence of a stationary distribution. However, not all MDPs have a stationary distribution for every policy π. For systems that do have a stationary distribution, steady-state behavior might not be

[2] A sketch of the proof is given in Appendix 3.B.

[3] The dual and its partial derivatives and Hessians are given in Appendix 3.A.

realizable for real systems that need to be started and stopped. Furthermore, transient behavior, such as the swing-up of a pendulum, might be of greater interest than steady-state behavior.

We can ensure the system has a stationary distribution that includes such transients by resetting the system with a probability $1-\gamma$ at each time step. The system is subsequently set to a state from the initial state distribution $p_1(\mathbf{s})$. In this case, the expected value of V at the next time step is given by

$$\begin{aligned}\mathbb{E}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}] &= \int_{\mathcal{S}} \gamma \mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'} V(\mathbf{s}') + (1-\gamma) p_1(\mathbf{s}') V(\mathbf{s}') \mathrm{d}\mathbf{s}', \\ &= \gamma \boldsymbol{\alpha}^T \tilde{\mathbf{K}}_s \boldsymbol{\beta}(\mathbf{s},\mathbf{a}) + (1-\gamma)\boldsymbol{\alpha}^T \hat{\boldsymbol{\mu}}_{S1}, \end{aligned} \tag{3.14}$$

where $\hat{\boldsymbol{\mu}}_{S1}$ is the empirical (observed) embedding of the initial state distribution and $\mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'}$ are the transition probabilities of the original MDP. Such a reset procedure enables learning by removing the impracticable requirement of infinite roll-out length. In this way, we obtain a discounted setting similar to that used in RL methods that optimize the accumulated discounted reward.

3.2.4 Generalization of the Sample-Based Policy

The parameters resulting from the optimization, η and $\boldsymbol{\alpha}$, can be inserted back in Equations (3.12) and (3.6) to yield the probabilities $\{p_\pi(\mathbf{s}_1,\mathbf{a}_1),\ldots,p_\pi(\mathbf{s}_n,\mathbf{a}_n)\}$ at the sampled $(\mathbf{s},\mathbf{a})$ pairs (where $p_\pi(\mathbf{s},\mathbf{a}) = \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ as before). Conditioning on the current state yields the policy to be followed in the next iteration. However, since states and actions are continuous, we need to generalize from these weighted samples to nearby data points. To this end, we want to find a generalizing stochastic policy $\tilde{\pi}(\mathbf{a}|\mathbf{s})$ conditioned on the observed policy samples.

We first consider parametric policies $\tilde{\pi}(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})$ linear in features $\boldsymbol{\phi}(s)$ with parameters $\boldsymbol{\theta}$. Later, we will generalize our results to non-linear kernels. We place a Gaussian prior over the unknown policy parameters, and choose a Gaussian noise model for the conditional over actions. Consequently, a Bayesian model is specified that will allow us to find a posterior over parameters $\boldsymbol{\theta}$

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(0,\alpha^{-1}\mathbf{I}), \\ \tilde{\pi}(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}^T\boldsymbol{\phi}(\mathbf{s}),\beta^{-1}\mathbf{I}). \end{aligned}$$

By conditioning on the sampled state-action pairs, we obtain the familiar update equation

$$p(\boldsymbol{\theta}|\mathbf{a}_1,\ldots,\mathbf{a}_n,\mathbf{s}_1,\ldots,\mathbf{s}_n) = z^{-1}p(\boldsymbol{\theta})\prod_{i=1}^{n}\tilde{\pi}(\mathbf{a}_i|\mathbf{s}_i;\boldsymbol{\theta}) \quad (\mathbf{s}_i,\mathbf{a}_i)\sim p_\pi(\mathbf{a},\mathbf{s}).$$

Algorithm 1 Policy iteration with relative entropy policy search (REPS)

repeat
  generate roll-outs according to $\tilde{\pi}_{i-1}$
  minimize dual
    $\eta^*, \boldsymbol{\alpha}^* \leftarrow \arg\min g(\eta, \boldsymbol{\alpha})$ Eq. 3.11
  calculate Bellman errors for each sample
    $\delta_j \leftarrow \mathcal{R}_j + \boldsymbol{\alpha}^{*T}\left(\tilde{\phi}(\mathbf{s}'_j) - \tilde{\phi}(\mathbf{s}_j)\right)$ Eq. 3.13
  calculate the sample weights
    $w_j \leftarrow \exp(\delta_j/\eta^*)$ Sec. 3.2.4
  fit a generalizing policy
    $\tilde{\pi}_i(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mu(\mathbf{s}), \sigma^2(\mathbf{s}))$ Sec. 3.2.4
until convergence

Here, z^{-1} is a normalization factor. However, our samples are drawn from $q(\mathbf{a}, \mathbf{s})$ and not $p_\pi(\mathbf{a}, \mathbf{s})$. From a log transformation of the update equation, we see that we can use importance sampling to estimate $\boldsymbol{\theta}$ using the approximation

$$\begin{aligned}
&\log p(\boldsymbol{\theta}|\mathbf{a}_1, \mathbf{s}_1, \ldots, \mathbf{a}_n, \mathbf{s}_n) \\
&\qquad = \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log \tilde{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}) + \text{const.} && (\mathbf{s}_i, \mathbf{a}_i) \sim p_\pi(\mathbf{a}, \mathbf{s}), \\
&\qquad \approx \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \frac{p_\pi(\mathbf{a}, \mathbf{s})}{q(\mathbf{a}, \mathbf{s})} \log \tilde{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}) + \text{const.} && (\mathbf{s}_i, \mathbf{a}_i) \sim q(\mathbf{a}, \mathbf{s}), \\
&\qquad = \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log \tilde{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})^{\frac{p_\pi(\mathbf{a}, \mathbf{s})}{q(\mathbf{a}, \mathbf{s})}} + \text{const.} && (\mathbf{s}_i, \mathbf{a}_i) \sim q(\mathbf{a}, \mathbf{s}),
\end{aligned}$$

As the new state-action distribution p_π is of the form given in (3.6), we can write the importance weights

$$w_i = \frac{p_\pi(\mathbf{s}_i, \mathbf{a}_i)}{q(\mathbf{s}_i, \mathbf{a}_i} = \exp\left(\frac{\delta(\mathbf{s}_i, \mathbf{a}_i, V^*)}{\eta^*}\right).$$

Since $\tilde{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$ is Gaussian in our model, raising to the power of w_i simply scales the variance by $1/w_i$. By exponentiating both sides again, and using the familiar procedure for weighted Bayesian linear regression [Gelman et al., 2004], we find the predictive distribution

$$\tilde{\pi}(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\beta \boldsymbol{\phi}(\mathbf{s})^T \mathbf{S}_n \boldsymbol{\Phi}^T \mathbf{D}^{-1} \mathbf{A}, \boldsymbol{\phi}(\mathbf{s})^T \mathbf{S}_n \boldsymbol{\phi}(\mathbf{s}) + \beta^{-1}), \tag{3.15}$$

$$\text{where} \quad \mathbf{S}_n = (\beta \boldsymbol{\Phi}^T \mathbf{D}^{-1} \boldsymbol{\Phi} + \alpha \mathbf{I})^{-1},$$

where $\mathbf{D}$ is a diagonal weighting matrix with $\mathbf{D}_{ii} = 1/w_i$ and $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]^T$.

The policy mean is of the form of the lower bound introduced by Dayan and Hinton [1997], and the policy that maximizes this lower bound can be found through a weighted linear regression [Peters and Schaal, 2007], although that framework does

not employ a Bayesian formulation and therefore cannot represent uncertainty in the parameters. Note that instead of basing the weights on a transformation of the reward function, our approach uses a transformation of the Bellman error, which takes the long-term expected rewards into account. The critical step of choosing the transformation was done by manual design in earlier work [Peters and Schaal, 2007, Dayan and Hinton, 1997]), while here the transformation directly results from the optimization problem (3.1)-(3.4). Algorithm 1 on page 43 shows how the different steps of our approach fit together in the special case of linear value functions and policy, and using sampled outcomes to approximate the Bellman error. This form of the algorithm was introduced in earlier work [Peters et al., 2010].

3.2.5 Non-Parametric Generalizing Policies

For non-parametric policies, Eq. (3.15) can be kernelized straightforwardly, to yield

$$\tilde{\pi}(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mu(\mathbf{s}), \sigma^2(\mathbf{s})), \quad \mu(s) = \mathbf{k}_s(\mathbf{s})^T(\mathbf{K}_s + \lambda\mathbf{D})^{-1}\mathbf{A}, \tag{3.16}$$

$$\sigma^2(\mathbf{s}) = k(\mathbf{s},\mathbf{s}) + \lambda - \mathbf{k}_s(\mathbf{s})^T(\mathbf{K}_s + \lambda\mathbf{D})^{-1}\mathbf{k}_s(\mathbf{s}), \tag{3.17}$$

where kernel vector $\mathbf{k}_s(\mathbf{s}) = \boldsymbol{\phi}(\mathbf{s})^T\boldsymbol{\Phi}$, kernel matrix $\mathbf{K}_s = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$, and λ is a free regularization hyper-parameter. Together with other hyper-parameters, such as the kernel bandwidth, λ can be set by performing cross-validation on a maximum marginal likelihood objective.

Kober et al. [2011] derived a EM-based policy search approach that uses similar cost-sensitive Gaussian processes. The regularization term in these Gaussian processes is modulated with the inverse of the weight at each data point. However, there are some notable differences. First of all, in our case the weights w are found by transforming the advantage function rather than the reward function, allowing decision making in longer-horizon problems. Furthermore, that approach used a maximum likelihood perspective which allows derivation of the mean, but not the variance. In contrast, we derive our policy update using importance sampling from a Bayesian perspective that allows a principled derivation of the covariance update. Algorithm 2 shows how the different steps of our approach, non-parametric relative entropy policy search (NP-REPS), fit together.

3.2.6 Hyper-parameter Optimization

The computation of the conditional operator has open hyper-parameters, namely, the hyper-parameters of the kernels over $\mathbf{s}$ and $\mathbf{a}$ as well as the regularization parameter λ. We set λ and the hyper-parameters of kernel k_a through two-fold cross-validation on the objective

$$\sum_{i=1}^{n} \left\| \boldsymbol{\phi}(\tilde{\mathbf{s}}_i)^T\boldsymbol{\phi}(\mathbf{s}'_i) - \boldsymbol{\phi}(\tilde{\mathbf{s}}_i)^T\hat{\mathbf{C}}_{S'|S,A}\boldsymbol{\psi}(\mathbf{s}_i,\mathbf{a}_i) \right\|_2^2,$$

Algorithm 2 Policy iteration with non-parametric REPS (NP-REPS)

repeat
  generate roll-outs according to $\tilde{\pi}_{i-1}$
  calculate kernel embedding strengths
    $\boldsymbol{\beta}_j \leftarrow (\mathbf{K}_{\text{sa}} + \lambda \mathbf{I})^{-1} \mathbf{k}_{\text{sa}}(\mathbf{s}_j, \mathbf{a}_j)$ Sec. 3.2.2
  minimize kernel-based dual
    $\eta^*, \boldsymbol{\alpha}^* \leftarrow \arg\min g(\eta, \boldsymbol{\alpha})$ Eq. 3.11
  calculate kernel-based Bellman errors
    $\delta_j \leftarrow \mathcal{R}_j + \boldsymbol{\alpha}^{*T} \left(\tilde{\mathbf{K}}_{\text{s}} \boldsymbol{\beta}_j - \mathbf{k}_{\text{s}}(\mathbf{s}_j) \right)$ Eq. 3.12
  calculate the sample weights
    $w_j \leftarrow \exp(\delta_j / \eta^*)$ Sec. 3.2.4
  fit a generalizing non-parametric policy
    $\tilde{\pi}_i(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mu(\mathbf{s}), \sigma^2(\mathbf{s}))$ Sec. 3.2.5
until convergence

which minimizes the difference between actual embedding strength and the embedding strength predicted using the conditional operator $\hat{\mathbf{C}}_{S'|S,A}$ introduced in Equation (3.10). This objective is based on the cross-validation objective proposed by Grünewälder et al. [2012a], but exploits the fact that the embedding will only be evaluated at known functions $\boldsymbol{\phi}(\tilde{\mathbf{s}})$.

The hyper-parameters of k_{s}, the kernel for the predicted variable $\mathbf{s}'$, cannot be tuned this way as trivial solutions exist. For example, essentially constant $\boldsymbol{\phi}(\mathbf{s}')$ with very high bandwidth minimize the prediction error. Instead, we set the hyper-parameters of k_{s} through minimization of the mean squared TD error in a two-fold cross-validation procedure. We choose this objective since minimizing the (residual) TD error is a common objective for feature selection [Parr et al., 2008] in reinforcement learning.

For the Gaussian process policy, we optimize the kernel hyper-parameters independently. The employed optimization objective is the weighted marginal likelihood, with weights w_i as discussed in Sec. 3.2.5. This objective is maximized in a cross-validation procedure where every roll-out is used as separate fold.

3.2.7 Efficient Approximations for Large Data Sets

The proposed algorithm requires inverting several $n \times n$ matrices, where n is the number of samples. If open hyper-parameters (such as regularization parameters or kernel bandwidths) need to be optimized, these inversion happens inside the optimization loop. As a consequence, learning becomes slow if more than approximately 5000 samples are used on current computing hardware. Especially for complex problems and problems requiring a high control frequency, such a soft limit can be prohibitive. In order to scale our method to larger problems, approximate methods with high time efficiency have to be considered. In this section, we will discuss two families of methods: sparsification of the kernel matrix, and approximation of the kernel function using stochastic features. Related work on efficient calculations for non-parametric reinforcement learning are discussed in Section 3.4.4 on page 70.

Sparsification Approaches. One way of scaling up kernel methods is to consider sparsifications, where a small number of pseudo-inputs are used rather than the full data set. Multiple sparsification schemes have been proposed, notably the likelihood approximation used in the projected latent variables (PLV) approach [Seeger et al., 2003] and the Bayesian derivation of sparse pseudo-input Gaussian processes (SPGPs) by Snelson and Ghahramani [2006] in the context of supervised learning.

Such sparsifications have been used in a number of RL algorithms [Engel et al., 2003, Xu et al., 2014, Jung and Polani, 2007, Xu et al., 2007, Lever and Stafford, 2015]. Often, a quadratic program is optimized to learn the embedding strength of the data points in the active set m. This approach typically results in an approximation $k(\mathbf{x},\mathbf{x}') \approx \mathbf{k}_m(\mathbf{x})^T \mathbf{K}_{mm}^{-1} \mathbf{k}_m(\mathbf{x}')$. In this equation, $\mathbf{k}_m(\mathbf{x})$ and $\mathbf{K}_{mm}$ are a vector and a matrix, respectively, of similarities to the active set of m data-points [Jung and Polani, 2007, Xu et al., 2007]. The same effective kernel is used in the PLV approach [Seeger et al., 2003]. Note that, effectively, a non-stationary kernel is obtained that is parametrized by the data points in the active set m [Jung and Polani, 2007, Xu et al., 2007].

The SPGP approach uses a very similar kernel, but includes a state-dependent regularization term. This term proves to be helpful in gradient-based hyper-parameter optimization [Snelson and Ghahramani, 2006]. The covariance of output points is then given by $\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} + \lambda\mathbf{I} + \mathbf{\Lambda}$. In this equation, the n^{th} element on the diagonal of $\mathbf{\Lambda}$ is given by $k(\mathbf{x}_n,\mathbf{x}_n) - \mathbf{k}_m(\mathbf{x}_n)^T \mathbf{K}_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_n)$. The matrix $\mathbf{K}_{nm}$ denotes a Gram matrix between all n input points and the active subset of m data points, and k_m and $\mathbf{k}_m$ denote a scalar and vector of corresponding kernel values.

To derive a sparsification with the same type of effective kernel using the reward-dependent regularization terms introduced in Section 3.2.5, we can start with the regular update equation for the non-parametric policy mean in Equation (3.16),

$$\mu(\mathbf{x}) = \mathbf{k}_n(\mathbf{x})^T (\mathbf{K}_{nn} + \lambda\mathbf{D})^{-1}\mathbf{y},$$

where $\mathbf{y}$ is the vector of all training outputs. We replace the occurrences of the kernel with the effective kernel

$$\mu(\mathbf{x}) = \mathbf{k}_m(\mathbf{x})^T \mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}(\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} + \lambda\mathbf{D} + \mathbf{\Lambda})^{-1}\mathbf{y}.$$

Now, we can apply the Woodbury identify to obtain the update equation

$$\mu(\mathbf{x}) = \mathbf{k}_m(\mathbf{x})^T (\mathbf{K}_{mm} + \mathbf{K}_{mn}(\lambda\mathbf{D} + \mathbf{\Lambda})^{-1}\mathbf{K}_{nm})^{-1}\mathbf{K}_{mn}(\lambda\mathbf{D} + \mathbf{\Lambda})^{-1}\mathbf{y}.$$

The empirical conditional embedding (Eq. 3.10) can similarly be approximated using this effective kernel. Starting from the regular update equation for the non-parametric policy covariance in Equation (3.17), in similar fashion we can derive the predictive variance

$$\sigma^2(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}_m(\mathbf{x})^T (\mathbf{K}_{mm}^{-1} + (\mathbf{K}_{mm} + \mathbf{K}_{mn}(\lambda\mathbf{D} + \mathbf{\Lambda})^{-1}\mathbf{K}_{nm})^{-1})\mathbf{k}_m(\mathbf{x}) + \lambda.$$

Analogously, a cost-regularized version of the PLV approach can be obtained in a similar form, but omitting the Λ terms. As the proposed sparsification scheme depends on

the inverse of $\mathbf{K}_{mm}$, numerical problems can potentially ensue if the active subset m is poorly chosen such that two almost-equal data points are present. To address this issue, we regularize $\mathbf{K}_{mm}$ where necessary.

Alternatively, we consider fitting a regular Gaussian process to a set of M inducing inputs with pseudo-targets given by weighted linear regression. For pseudo-outputs $\tilde{\mathbf{y}}$ of M sparse inputs, a Gaussian process would predict $\hat{\mathbf{y}} = \mathbf{K}_{nm}(\mathbf{K}_{mm} + \lambda\mathbf{I})^{-1}\tilde{\mathbf{y}}$ at all (active and passive) inputs. Since we know the true outputs $\mathbf{y}$ for the training data, a maximum likelihood solution can be found using weighted linear regression, considering $\mathbf{K}_{nm}(\mathbf{K}_{mm} + \lambda\mathbf{I})^{-1}$ as design matrix. This approach yields the update equations for the mean

$$\mu(\mathbf{x}) = \mathbf{k}_m \tilde{\mathbf{K}}_{mm}^{-1} \tilde{\mathbf{y}},$$

where $\tilde{\mathbf{K}}_{mm}^{-1}$ is a regularized inverse $(\mathbf{K}_{mm} + \lambda\mathbf{I})^{-1}$. The inducing output values $\tilde{\mathbf{y}}$ are

$$\tilde{\mathbf{y}} = (\tilde{\mathbf{K}}_{mm}^{-1}\mathbf{K}_{mn}\mathbf{D}\mathbf{K}_{nm}\tilde{\mathbf{K}}_{mm}^{-1} + \lambda_2\mathbf{I})^{-1}\tilde{\mathbf{K}}_{mm}^{-1}\mathbf{K}_{mn}\mathbf{D}\mathbf{y},$$

where λ_2 is an additional regularization parameter. As the covariance does not depend on the inducing output at the training points, the standard update equation for Gaussian processes can be used in this case. The update equation for the mean corresponds to that of a Gaussian process with an effective kernel $k(x, x') = \mathbf{k}_m(x)^T \tilde{\mathbf{K}}_{mm}^{-1} \tilde{\mathbf{K}}_{mm}^{-1} \mathbf{k}_m(x')$.

Random Fourier Features. Instead of sparsification, that is, using the exact kernel function at a subset of data points, we might instead approximate the kernel function at all data points. One such approach is proposed by Rahimi and Recht [2007], who define a distribution $p(z)$ over mapping functions z such that the inner products in sampled feature spaces are unbiased estimates of the kernel evaluation $k(\mathbf{x}, \mathbf{y}) = \mathbb{E}[z(\mathbf{x})^T z(\mathbf{y})] \approx \mathbf{z}(\mathbf{x})^T \mathbf{z}(\mathbf{y})$, where $\mathbf{z}_i(\cdot) \sim p(z)$. They propose two kinds of random features that obey this criterion, Fourier features and binning features. As the binning features are suitable only for kernels that solely rely on the L1 norm between data points, we will focus on the Fourier features in this section. These features have been used successfully in various classification and regression tasks [Rahimi and Recht, 2007, Hernández-Lobato et al., 2014, Lu et al., 2016].

For the Fourier features, we require the Fourier transform $\alpha p(\boldsymbol{\omega})$ of a stationary (shift-invariant) kernel $k(\mathbf{x}, \mathbf{y})$, where α is a normalization factor that ensures $p(\boldsymbol{\omega})$ is a probability distribution. The inverse Fourier transform is thus

$$k(\mathbf{x}, \mathbf{y}) = \alpha \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{y})} \mathrm{d}\boldsymbol{\omega} = 2\alpha\mathbb{E}[\cos(\boldsymbol{\omega}^T\mathbf{x} + b)\cos(\boldsymbol{\omega}^T\mathbf{y} + b)],$$

where $b \sim \mathcal{U}(0, 2\pi)$. We can approximate this integral using samples $(\boldsymbol{\omega}_i, b_i)$ with $i = 1, \ldots, L$ and obtain

$$k(\mathbf{x}, \mathbf{y}) \approx 2\alpha \frac{1}{L} \sum_{i=1}^{L} \cos(\boldsymbol{\omega}_i^T\mathbf{x} + b_i)\cos(\boldsymbol{\omega}_i^T\mathbf{y} + b_i) = \mathbf{z}(\mathbf{x})^T\mathbf{z}(\mathbf{y}),$$

with $\mathbf{z}_i(\mathbf{x}) = \sqrt{2\alpha L^{-1}}\cos(\boldsymbol{\omega}_i^T\mathbf{x} + b_i)$. As the chosen number of samples L gets smaller, the approximation gets coarser but computations will get faster, as we will need to invert $L \times L$ covariance matrices.

In this article, we will use the squared-exponential (or Gaussian) kernel $k_{\text{es}}(\mathbf{x},\mathbf{y}) = \alpha\exp(-0.5||\mathbf{x}-\mathbf{y}||_{\Sigma^{-1}})$ with diagonal covariance Σ. This kernel has a corresponding Fourier transform $p_{\text{es}}(\boldsymbol{\omega}) = \mathcal{N}(0,\Sigma)$. Furthermore, for periodic data we will use a periodic kernel $k_{\text{p}}(\mathbf{x},\mathbf{y}) = \alpha\exp(-2\sin^2(0.5|\mathbf{x}-\mathbf{x}'|)/\sigma^2)$. This kernel is equivalent to the Gaussian kernel on periodic features

$$k_{\text{p}}(\mathbf{x},\mathbf{y}) = k_{\text{es}}([\cos(\mathbf{x}),\sin(\mathbf{x})]^T,[\cos(\mathbf{y}),\sin(\mathbf{y})]^T).$$

Therefore, suitable random features can be generated as

$$\mathbf{z}_i(x) = \sqrt{2\alpha L^{-1}}\cos([\cos(\mathbf{x}),\sin(\mathbf{x})]\boldsymbol{\omega}_i + b_i).$$

Products of Gaussian and periodic kernels can likewise be written as a single multivariate Gaussian kernel on appropriate features of the inputs and handled in a similar way. As a result, all matrix inversions are now performed on $L \times L$ rather than $n \times n$ matrices.

3.2.8 Feature Learning for High-dimensional Noisy Sensors

The proposed non-parametric relative entropy policy search method relies on distances between input points. In high dimensional sensor spaces, low-intensity noise or small illumination changes may cause large displacements, making distances less meaningful.

Auto-encoders [Vincent et al., 2008, Bengio, 2009, Kingma and Welling, 2013] are neural networks that have been shown to successfully learn meaningful low-dimensional representations of (robot) movement data [Chen et al., 2015, Mattner et al., 2012, Finn et al., 2016]. Therefore, we propose using the representation learned by such auto-encoders as input for reinforcement learning of policies of non-task specific form.

Conventional auto-encoders attempt to reconstruct the input signal from a low-dimensional representation. Such representations can be used in reinforcement-learning tasks [Lange et al., 2012, Mattner et al., 2012]. However, this reconstruction objective could be increased by either reconstructing task-relevant signal variations, or task-irrelevant influences. Therefore, we employed a variant of the variational auto-encoder described by van Hoof et al. [2016a]. This variant is trained using the objective of predicting the input signal from the previous input signal and the executed action. As such, the encoder cannot improve its performance by reconstructing high-frequency noise, as such noise is not predictive of future signals.

Similar to earlier studies [Finn et al., 2016, Watter et al., 2015], this approach aims at learning representations that respect the temporal dynamics of the task. Whereas Finn et al. [2016] used a separate exploration controller using a simple state and reward function excluding the high-dimensional sensor space, we aim to

learn feedback policies directly from the high-dimensional space. Compared to Watter et al. [2015], who learned control policies in a single shot based on data under an exploration policy, we aim to learn iteratively on-policy. As the policy starts to generates more relevant samples, the learned feature representation can be improved.

3.3 Experiments

We evaluate our method on a reaching task, two variations of the underpowered pendulum swing-up task, and a real-robot stabilization task with high-dimensional tactile input. For the underpowered pendulum swing-up task, we consider a standard version of the swing-up task where the agent has access to the angle θ and angular velocity $\dot{\theta}$ directly. In a second version we use a real robot that has access only to camera images, resulting in a high-dimensional input space. In this section, we will first discuss elements of our setup that are the same across tasks. Subsequently, we discuss implementation specifics and results for each tasks separately.

3.3.1 Experimental Setup

We assume a realistic exploration setup in which the agent cannot choose arbitrary state-action pairs. Instead, as shown in Alg. 2, from an initial state distribution our agent explores using its stochastic policy. After every 10 roll-outs, the model learner and the policy of the agents are updated. To bootstrap the model and the policy, the agent is given 30 roll-outs using a random exploratory policy initially. To avoid excessive computations, we include a simple forgetting mechanism that only keeps the latest 30 roll-outs at any time[4]. As each roll-out contains 49 time steps on average in the larger tasks (as episodes are reset with a constant probability after each step), most computations are performed on approximately 1500×1500 matrices.

After each update, in simulated runs, the learning progress is evaluated by running the learned policy on 100 roll-outs with a fixed random seed. This data is not used for learning. For every method, we performed 10 trials, each consisting of 20 iterations so that 220 roll-outs were performed per trial (30 initial roll-outs plus 10 per iteration). In real-robot experiments we evaluated the learned policy on the training samples. Exact settings for those experiments are given in the respective description. The model and policy are refined incrementally in every iteration.

3.3.2 Compared Methods

We compared learning progress of the proposed approach to that of various other approaches. On the one hand, we consider the non-parametric value-function based methods introduced by Grünewälder et al. [2012b] and Pazis and Parr [2011]. We also compare to versions of REPS that use the sample-based model approximation

[4] As a consequence, the reference distribution q is a mixture of the previous three state-action distributions in our experiments.

introduced by Daniel et al. [2016a], and one that uses a fixed feature set. The relationship of the proposed approach to earlier non-parametric reinforcement learning methods will be discussed in Section 3.4.3 on page 69.

Sample Based Model. REPS only needs $\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}]$ at observed state-action pairs $(\mathbf{s}_i,\mathbf{a}_i)$. Therefore, if the system is deterministic, this expectation is simply $V(\mathbf{s}'_i)$ at the observed values for $(\mathbf{s}_i,\mathbf{a}_i)$. In stochastic systems, this sample-based method is used as approximation.

Feature Based REPS with Fixed Basis Functions. Instead of the non-parametric form of V assumed in this chapter, we can follow earlier work and define a fixed feature basis [Peters et al., 2010, Daniel et al., 2016a]. We choose to use a similar number of the same radial basis functions used in the non-parametric method, but distribute these according to a regular grid over the state-action space.

Approximate Value Iteration. In this approach by Grünewälder et al. [2012b], the value function is assumed to be an element of the chosen RKHS. The maximization of the Q function requires discretizing $\mathbf{a}$, for which we choose 25 uniform bins in the allowable range. A deterministic policy selects $\mathbf{a}^* = \arg\max Q(\mathbf{s}_i,\cdot)$, but this policy does not explore, yielding bad performance in on-policy learning. To obtain an exploration-exploitation trade-off we replace the maximum by the soft-max operator $\mathbf{a}^* \propto \exp(Q(\mathbf{s}_i,\cdot)c/\mathrm{stdev}(Q(\mathbf{s}_i,\cdot)))$. The free parameter c specifies the greediness of the exploration/exploitation trade-off on normalized Q values. In an **on-policy** scheme, the new policy is used to obtain samples for the next iteration.

As a comparison to this on-policy scheme, we also compare using a **grid** of state-action pairs as training data. For a dense grid, this method has a richer input than all other methods, as those other methods start with uninformed roll-outs from the initial-state distribution. In a real system, obtaining such a grid is often unrealistic as the state cannot be set arbitrarily.

Non-parametric Approximate Linear Programming. Pazis and Parr [2011] introduce a non-parametric method, NPALP, that assumes the value function is Lipschitz. This assumption allows the RL problem to be formalized as a linear program. This method assumes the dynamics are deterministic. A greedy policy is obtained that is optimal if all state-action pairs have been visited. Since visiting all state-action pairs is infeasible in continuous systems, we include exploration by adding Gaussian distributed noise to the action in a fraction ϵ of the selected actions.

3.3.3 Approximation Methods

We compare the approximation methods described in Sec. 3.2.7 to each other as well as to a naive baseline. This comparison is done across a range of different numbers of features and/or inducing inputs, to evaluate the trade-off between a better approximation quality and a faster computation time.

Sub-sampling as Baseline. For the sub-sampling baseline, we simply train the method as described in Section 3.2.2 with a subset of the data points, ignoring the points that are not in the random subset.

Sparse Pseudo-input Gaussian Processes (SPGP). As an approximation, we use the method proposed by Snelson and Ghahramani [2006] to approximate the Gaussian process policy. To incorporate the desirability of data points, we use the modifications proposed in Section 3.2.7. As there is no straightforward extension of this method to learn the Bellman error terms (Equation 3.12), in these steps of the algorithm we interpret the matrix $\mathbf{K}_{nm}$ introduced in Section 3.2.7 as a feature matrix and subsequently calculate the Bellman error terms using the feature-based formulation introduced by Peters et al. [2010]. In contrast to that work, we do not use single sample roll-outs, but calculate the embedding strengths $\boldsymbol{\beta}$ (Sec. 3.2.2) using a linear kernel with the pseudo-features as input. Snelson and Ghahramani [2006] suggest choosing the active subset by maximizing the marginal likelihood. However, this maximization is a computationally intensive process, and for the value function approximation, there is no criterion available equivalent to the maximum likelihood criterion for the supervised learning setup. Therefore, we choose random subsets when applying this method.

Projected Latent Variables (LPV). Additionally, we compare to the sparsification method proposed by Seeger et al. [2003]. Like for the previous method, we interpret the $\mathbf{K}_{nm}$ matrix as features and use these to calculate embedding strengths and Bellman error terms.

Regression-based Sparse GPs. We also consider the regression-based sparse Gaussian Processes proposed in Section 3.2.7. Again, the $\mathbf{K}_{nm}$ matrix is interpreted as design matrix for calculation of the embedding strengths and Bellman error terms.

Fourier Transform Based Approximation. The last method we consider is based on the work of Rahimi and Recht [2007], with the extension to desirability-weighted samples as described in Section 3.2.7. As this method approximates the policy using a features, we can use these features to calculate embedding strengths and Bellman error terms in the feature-based formulation introduced by Peters et al. [2010].

3.3.4 Reaching Task Experiment

In the reaching task, we simulate a simple two-link planar robot. In this task, the agent's actions directly set the accelerations of the two joints. Each link is of unit length and mass, and the system is completely deterministic. The agent gets negative reinforcement $r(\mathbf{s}, \mathbf{a}) = -10^{-4}||\mathbf{a}||_2^2 - ||\mathbf{x} - \mathbf{x}_{\text{des}}||_2^2$ according to the square of the applied action and of the distance of its end-effector to the Cartesian position $\mathbf{x}_{\text{des}} = [0.5, 0]$. Note that actions are two dimensional and states are four dimensional (joint positions and velocities). The robot starts stretched-out with the end-effector at $\mathbf{x} = [0, 2]$. The

maximum applied acceleration is 50ms^2. We use $\gamma = 0.96$, which resets roll-outs after 24 samples, on average.

We use the commonly used squared-exponential (or Gaussian) kernel for angular velocities $\dot{\boldsymbol{\theta}}$ and actions. This kernel is defined as $k_{\text{se}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_j))$, with $\mathbf{D}$ a diagonal matrix containing free bandwidth parameters. However, for the angles $\boldsymbol{\theta}$ we need a kernel that represents its periodicity. We chose the kernel $k_{\text{p}}(x_i, x_j) = \exp(-\sum_d \sin((x_i^{(d)} - x_j^{(d)})/(2\pi))^2/\mathbf{l}_d^2)$, where $\mathbf{l}$ is a vector of free bandwidth parameters. Consequently, we obtain a complete kernel

$$k((\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i, a_i), (\boldsymbol{\theta}_j, \dot{\boldsymbol{\theta}}_j, a_j)) = k_{\text{p}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) k_{\text{se}}(\dot{\boldsymbol{\theta}}_i, \dot{\boldsymbol{\theta}}_j) k_{\text{se}}(\mathbf{a}_i, \mathbf{a}_j),$$

composed of three simpler kernel functions. Since k_{p} is equivalent to a squared-exponential kernel over the sine and cosine of the angle (Sec. 3.2.7), the composite kernel is equivalent to single squared exponential kernel over composite vectors $[\cos\boldsymbol{\theta}, \sin\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \mathbf{a}]^T$.

Feature-based REPS needs a grid over the complete state-action space. Due to difficulties with the six-dimensional state-action space, we omitted this method. The exploration parameter ϵ of the NPALP method was set to 0.1, with the standard deviation of Gaussian noise set to $30Nm^2$. The Lipschitz constant was set to 1 with the velocity dimensions scaled by 1/5 for calculating distances. The exploration parameter c of the approximate value iteration method was set to 1.5. These values were manually tuned to yield fast and consistent learning progress. For REPS, we use a KL-bound ϵ of 0.5 in our experiments, as this value empirically yielded acceptably fast learning progress while keeping updates smooth enough on a range of learning problems.

Results of the Reaching Task. The results of the reaching task are shown in Figure 3.1. As this task is deterministic, the sample-based model is optimal and provides a upper bound to the performance we can expect to get. Since non-parametric REPS with model learning needs to iteratively learn the transition distribution, its convergence is slower. After inspection of individual trials, the wide variance seems to be caused by occasional failures to find good hyper-parameters for the state-action kernel.

The baseline methods NPALP and value iteration using RKHS embeddings obtain good performance after the couple of iterations. However, if we iterate performing roll-outs and policy updates after those first steps, these methods fail to improve the policy consistently. The grid based value iteration scheme fails in this case: due to memory limitations the maximum grid size we could use was $[5 \times 5 \times 5 \times 5 \times 2 \times 2]$ in the 6-dimensional space, which appears to be insufficient to learn the task.

3.3.5 Low-Dimensional Swing-up Experiment

In this experiment, we simulate a pendulum with a length of $l = 0.5$m and a mass $m = 10$kg distributed along its length. A torque a can be applied at the pivot. The pendulum is modeled by the dynamics equation $\ddot{\theta} = (glm\sin\theta + a - k\dot{\theta})/(ml^2/3)$,

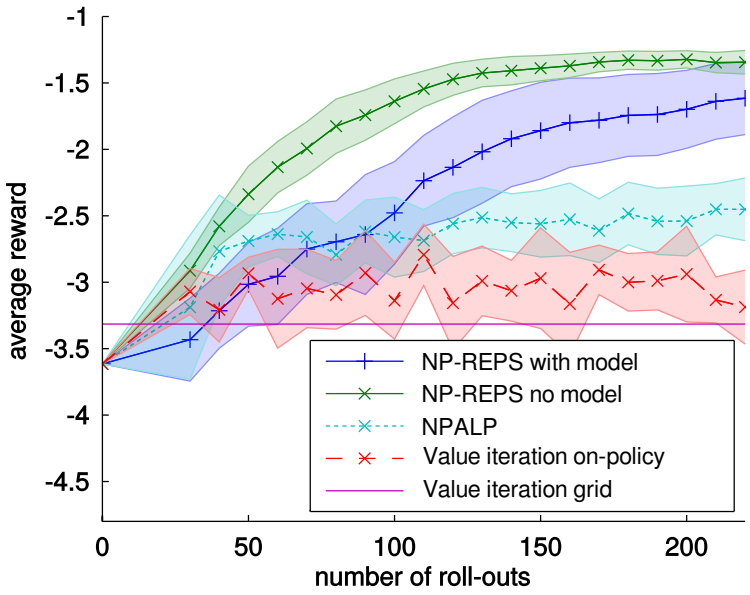


Figure 3.1.: Comparison of learning progress of different methods on the reaching task. As this environment is deterministic, sample-based approximations of the transition functions are optimal. Error bars show twice the standard error of the mean. The value-iteration method does not depend on roll-outs, its performance is shown for comparison.

where $k = 0.25$Ns is a friction coefficient and $g = 9.81$ is the gravitational constant. The controller's sampling frequency is 20 Hz, i.e., every 0.05s the agent gets a reward and chooses a new action. The maximum admissible torque is 30Nm, which prevents a direct swing-up from the downwards position. Additive noise with a variance of $1/dt$ disturbs the controls, resulting in a standard deviation of about 4.5Nm per time step. The reward function was set to $r(s,a) = -10\theta^2 - 0.1\dot{\theta}^2 - 10^{-3}a^2$, where θ is mapped to $[-0.5\pi, 1.5\pi)$ to differentiate the rewards of clockwise and counterclockwise swing-ups. We use a reset probability of 0.02 ($\gamma = 0.98$).

The algorithms directly access the state variable encoding the angle θ and the angular velocity $\dot{\theta}$, i.e., the state is defined as $\mathbf{s} = [\theta, \dot{\theta}]^T$. The same kernels are used as in the reaching task experiment (Sec. 3.3.4). The NPALP method was not designed for stochastic systems and is consequently omitted. We set the grid size for feature-based REPS to $[10 \times 10 \times 10]$ and for the value-iteration method to $[19 \times 11 \times 11]$. The greediness parameter c for on-policy value-iteration was set to 2 after manual tuning. The KL bound ϵ was again set to 0.5.

On this task, we also compare the different methods for approximating the kernel matrix. We do this by training policies using the various methods discussed in Section 3.3.3. The trade-off between accuracy and computational speed is defined by the number of inducing inputs for the sparse kernel methods, or by the number of random features used in the Fourier-transform based approximation. The number of features (respectively, inducing inputs) was varied to investigate the trade-off between approximation accuracy and computation speed. We measured the learning progress using the average reward, and additionally logged the time needed per iteration for each

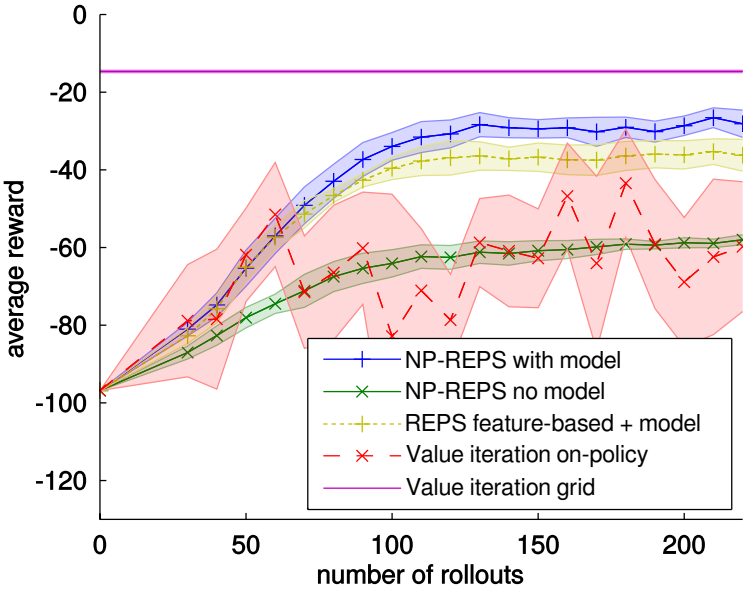


Figure 3.2.: Results of the low-dimensional swing-up experiment. Comparison of learning progress of different methods on the swing-up task. Our non-parametric relative entropy method outperforms the other on-policy learners. Error bars show twice the standard error of the mean.

condition. We report the average time needed over 10 trials and 10 iterations per trial.

Results of the Low-Dimensional Swing-up. We show a comparison of the discussed methods in Figure 3.2. The value iteration methods starts out competitively, but fails to keep improving the policy. Large variance indicates the learning process is unstable. The bounded policy update in REPS makes learning progress smooth by limiting information loss, and trades off exploration and exploitation.

The sample-based baseline model performs considerably worse in this experiment, as it cannot account for stochastic transitions. The variant with fixed features performs well initially, but in later iterations, the non-parametric method is able to focus its representative power on frequently visited parts of the state-space, resulting in an improved performance.

In contrast to the previous experiment, here, the grid-based value iteration method worked well. The policy learned by NP-REPS, shown in Figure 3.3, sometimes overshoots the inverted position, which the grid based value iteration avoids. However, to reach this performance, a grid of training samples covering the full state-action space was needed. Providing such a grid is only feasible in simulation, as without an existing controller, it is generally not possible to start the dynamical system with arbitrary position and velocity. Furthermore, on higher-dimensional tasks, a grid of sufficient resolution would require impracticably many samples.

Results and Discussion of the Approximation Experiment. A comparison of our results using different approximation schemes and different numbers of features is

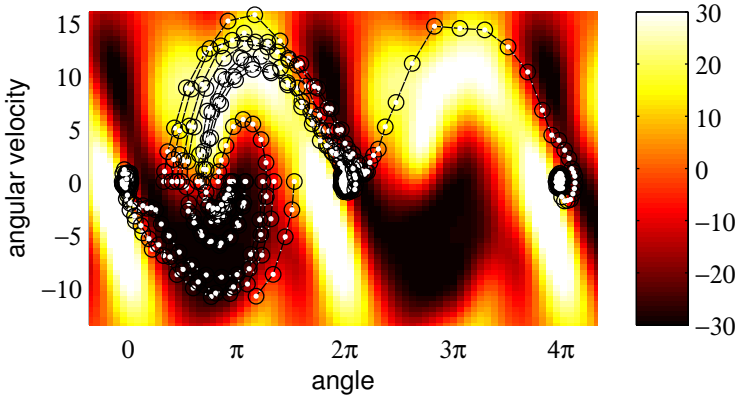


Figure 3.3.: Results of the low-dimensional swing-up experiment. Mean of the learned stochastic policy. Overlayed are 15 trajectories starting at the x-axis between 0 and 2π. Most roll-outs reach the desired inverted pose, possibly after one swing back and forth. One roll-out overshoots and makes a full rotation before stabilizing the pendulum.

shown in Figure 3.4. In Figure 3.4a, all available inputs were used, and hence no approximation is needed. Thus, the sub-sampling method reduces to the original methods, and performs best. However, if the number of available bases is reduced to 500 (about one third of the available training points), just training on a sampled subset performs very badly, as shown in Figure 3.4b. In this setup, the approximation based on a Fourier transform of the kernel seems the most suitable, both in terms of sample efficiency and of asymptotic value.

In Figure 3.4c, the available number of bases is reduced even more, to about 7% of the available training data. In this case, the regression-based sparsification method allows faster learning than the Fourier-based methods and both other sparsification methods, possibly because we used induced inputs $\tilde{y}$ that were optimal in a least-square sense. Considering that only 7% of the basis is available, the drop in performance relative to the full availability is modest.

Overall, the SPGP and LPV sparsification methods perform very similarly and yield similar asymptotic values as the regression-based sparsification methods. The main difference between these methods is the additional regularization term based on the approximation accuracy that Snelson and Ghahramani [2006] introduced. One of the main benefit of this additional regularization is a better behavior of the marginal likelihood for the purpose of optimizing the inducing input points [Snelson and Ghahramani, 2006]. As we did not optimize for these points (as that would be too computationally costly to do inside the reinforcement-learning loop), our results seems plausible in this respect.

An indication of the time requirement of the different methods is given in Figure 3.4d. The implementations of the algorithms are not directly comparable, so drawing hard conclusions about the algorithms is not possible on the basis of this data. One implementation issue is that cross-validation for the subsampled and Fourier-based methods can be implemented using a fast decremental update of the matrix inverse. For the other sparsification methods, there is no straightforward way to implement this speed-up so these were trained using a 2-fold cross-validation setup

(rather than using one fold for each trajectory, which would have been even more expensive computationally). Therefore, these three methods ('SPGP', 'LPV', and 'Sparse') are slowest in cases with many available inducing inputs.

When all data is available, applying any approximation takes more time than the baseline 'sampling' method. However, the 'Fourier' method tends to be much faster as the number of features is reduced—one of the reasons for this effect is that in our implementation, calculated features are cached. This benefits feature-based approximations, but not kernel-based ones. Compared to using all available bases without approximating, the 'Fourier' method with 500 bases is about 2 times as fast at a similar level of performance.

3.3.6 Real-Robot High-Dimensional Swing-up Experiment

In this experiment, we aim to validate our method on a real-robot task. Furthermore, the learner will not have direct access to joint angles or velocities, but only to high-dimensional image-based representations of the state obtained through a camera pointed at the robot.

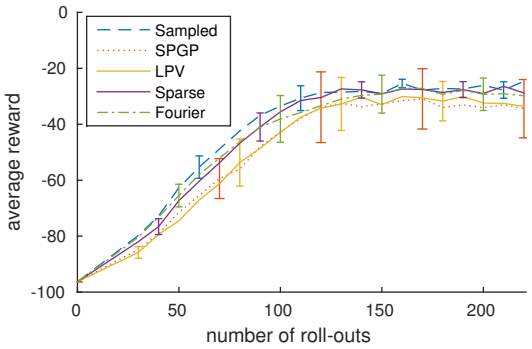
Robot Setup and System Dynamics. To the end-effector of a Kuka light-weight robot arm, we attached a wooden rod roughly 1kg heavy and 80 cm long. Red cardboard was used to make the pendulum visually salient. We aim to swing this rod up using the last degree-of-freedom, and limit the torque of the corresponding motor such that the pendulum cannot be swung up directly from the downwards position. This setup is shown in Figure 3.5a.

A couple of characteristics of the robot system make the task different from the simulated pendulum swing-up task in the previous section. The robot link is limited in its range of motion to less than 2π and in its velocity to less than about $4.2/s$, whereas the simulation had no such limitation. To prevent the robot from moving hard into these limits, we add a feedback signal that stops the robot if it gets close to those limits for safety[5]. The joint limits are illustrated in Figure 3.5b.

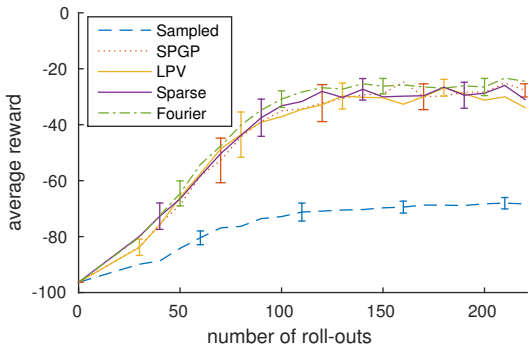
To perform exploration on the real robot, another issue to address is that high jerk might damage the gearboxes. Therefore, instead of controlling the torque as in the previous experiments, we will control an increment to the torque. This increment will smoothly be added to the previous torque over the course of one time step. Controlling the increment, rather than directly controlling the torque, has the additional benefit that the applied torque tends to be more consistent over time, preventing 'washing out' of high-frequent control signals on the robot system. To preserve Markov properties, the previous torque has to be appended to the state vector. To provide comparable results over the different robot trials, instead of resetting the system randomly, we reset the system every 50 time-steps to a set of ten different starting angles. The starting velocity and acceleration are set to zero.

Camera Setup and Image Processing. The camera provides video frames at a rate of about 30Hz, but this rate can vary slightly during the experiments. To prevent

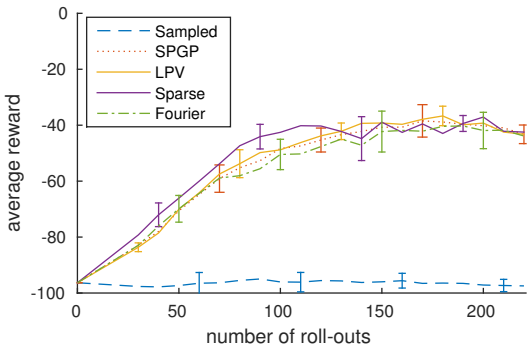
[5] The exact feedback signals used are given in Appendix 3.C.



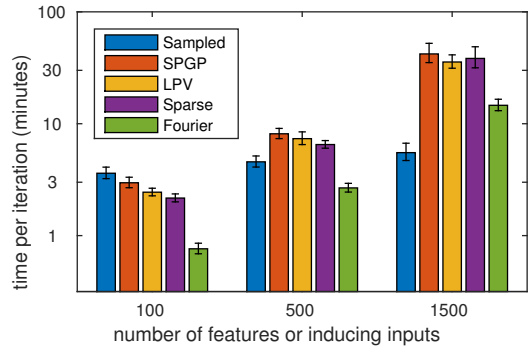
(a) Different approximation methods using 1500 features or all available (on average 1500) inducing inputs.



(b) Different approximation methods using 500 features or inducing inputs.



(c) Different approximation methods using 100 features or inducing inputs.



(d) Time requirement of different approximation methods on a 2.7 GHz processor running in single threads, log scale.

Figure 3.4.: Results of the evaluation of different approximation methods using different numbers of features. In all graphs, error bars indicate twice the standard error of the mean. The learning curves show error bars only for every fifth iteration to keep the figures interpretable.
At a medium number of features (b), the Fourier-based random features deliver performance that is almost equal to the original algorithm (a), while requiring fewer computational resources (d). More computational resources can be saved by using only 100 features or inducing inputs, at the cost of a larger performance gap to the original algorithm (c).

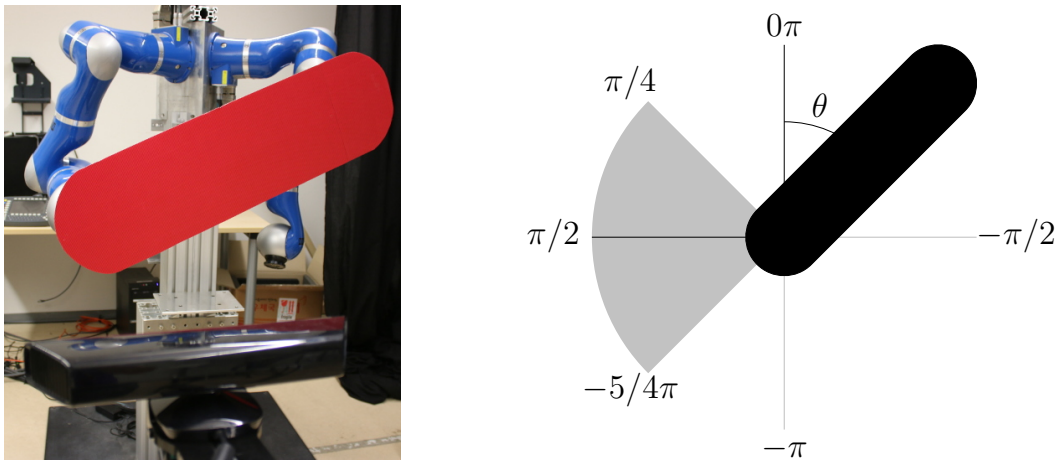


Figure 3.5.: The setup of the real-robot swing-up experiment with high-dimensional state representation. (a) The robot setup. One of the robot arms holds a pendulum which has a mass of about 1 kg and a length of about 80 cm. The Kinect camera in the foreground is used to provide feedback to the robot (proprioceptive joint information is not available to the robot). (b) An illustration of the setup that illustrates the coordinate system used. As the image is shown from the camera's point of view, the coordinate system is inverted (clockwise θ). In the shaded area, an additional torque is applied to keep the robot from running into the joint limit at $\pi/2$.

synchronization issues between the camera and the control, we let the arrival of camera images govern the time step length. Since we want to control the robot at about ten Hertz, we choose a new action whenever every third camera image is received. Consequently, not all torques are applied for the same duration, providing a source of transition noise.

To keep the amount of storage space and processing time limited, we down-sample the images to 15×20 pixels and reduce the image to a single channel (by subtracting the average brightness from the red channel value, since the color of the pendulum is red). The image is blurred slightly with a low-pass filter to smooth out sensor noise, using a Gaussian kernel with a bandwidth of 6% of the image width. To provide the learner with a notion of velocity, the 15×20 pixel image and the 15×20 difference image to the previous time-step were given to the robot as a concatenated 600-dimensional feature vector, as shown in Figure 3.6.

Employed Reward Function, Kernel, and Experimental Settings. In contrast to the earlier experiment, the real robot system is not periodic as it cannot turn the joint more than 2π. Consequently, we need a reward function that punishes the robot for getting too close to the joint limit. We use the reward function

$$r(\mathbf{s}, a) = \exp\left(-\frac{9\theta^2}{4}\right) - 2(\theta - 0.25)\,\mathbb{1}_{\{x \in \mathbb{R}: x > 0.25\}}(\theta) - 0.0005a^2,$$

where the first term rewards the robot for being close to the upright position and the second term punishes the robot for being too close to the joint limit. All else

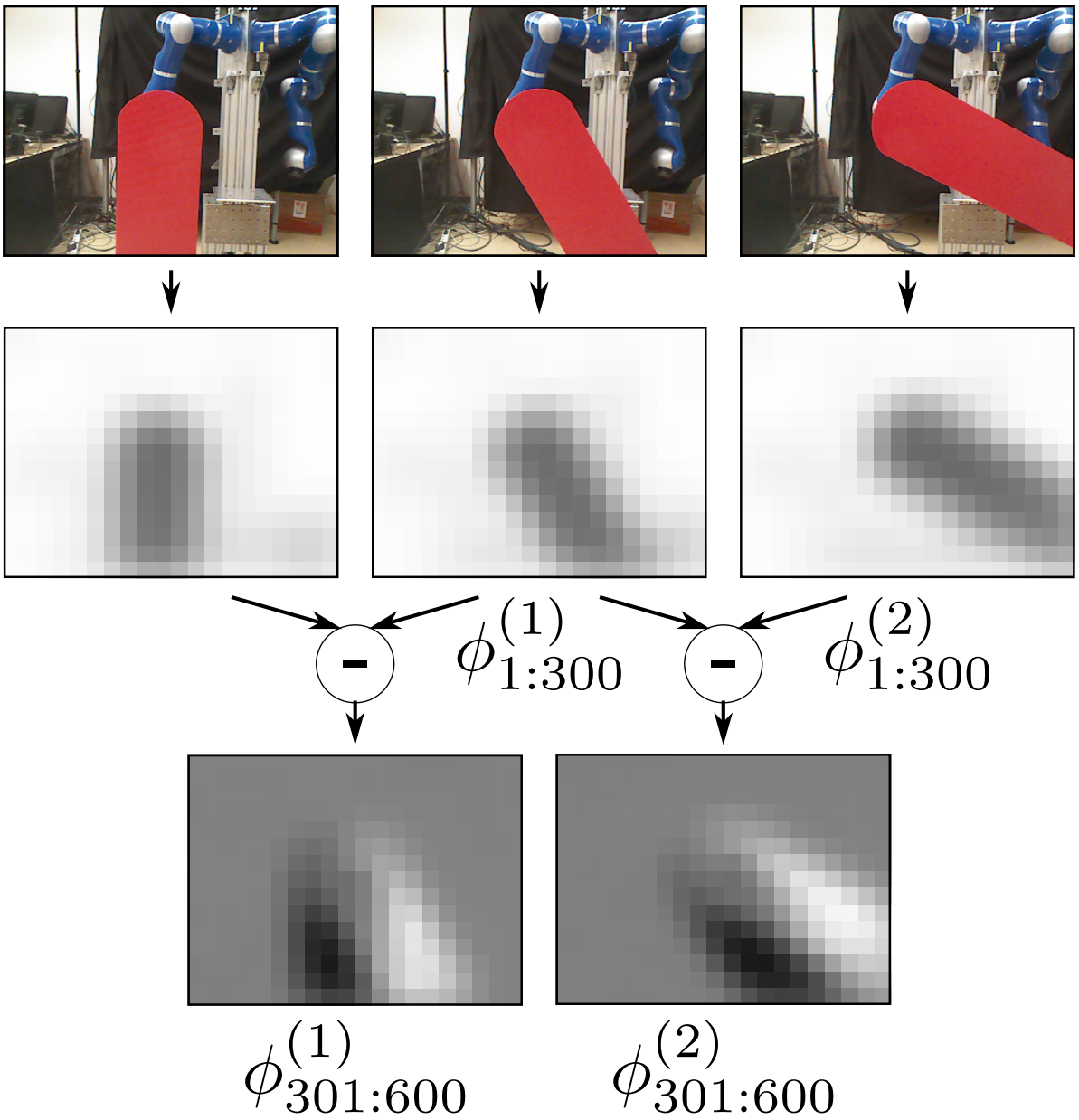


Figure 3.6.: Sensor representation used by the robot. The kinect camera image is down-sampled into 15 by 20 pixels and converted to a single-channel image by subtracting the average intensity from the red channel values. From the resulting sequence of images, the current image and the difference between the current image and the previous image are concatenated with the previously applied torque, yielding a 601-dimensional feature vector. The resulting real-robot reinforcement learning problem with high-dimensional state representations is extremely challenging for many current reinforcement learning methods.

being equal, we prefer solution that do not needlessly change the torques suddenly, as such behavior causes high jerk that can damage the robots. Therefore, the third term punishes large control actions.

As kernel on the state variables, we used a Gaussian kernel with three separate bandwidth parameters: one for all pixels of the current image, one for all pixels of the difference image, and one for the previously applied torque. The bandwidth parameters for all pixels of each image were tied together in this way to keep the hyper-parameter optimization manageable. For the learned transition model, the kernel on the action (applied jerk) is again a squared exponential.

In this experiment, the heuristic for setting the bandwidth for the value function approximation according to the TD-error did not always work. Therefore, these three bandwidth parameters were set by hand. The bandwidth parameters and other kernel parameters for the learned transition model and the policy were set automatically by maximizing the same cross-validation objectives used in the previous experiments. The bound on the KL divergence was set to $\epsilon = 0.8$. We performed 6 trials of 10 iterations each.

We used the Fourier feature approximation to the full kernel matrix, as the simulation experiments showed these features to yield good performance with an intermediate number of bases while also requiring relatively little computation time. Furthermore, computing the features and evaluating the linear Bayesian policy is straightforward to implement on a robotic system and can run in real-time as it has relatively low computational demands. We used 1000 random basis features, as the system is intrinsically higher dimensional compared to the simulated pendulum swing-up experiment (as we appended the previously applied torque to the state vector).

Sometimes, a feasible solution to the dual optimization problem could not be found. We considered this effect could be due to over-fitting to the sensor noise in the camera that made the system non-Markov. We addressed this issue by projecting the 1000-dimensional feature vectors on all principal components with principal values at least 5% of the maximum principal value for purpose of optimization of the dual function (Eq. 3.11) only. This procedure yielded between 100 and 200 components on our dataset and addressed the problem satisfactorily. The dimension reduction also sped up the optimization of the dual function. Models and policies were still learned in the original 1000-dimensional spaces, as these steps contain regularization terms that make them robust to such over-fitting.

Applicability of Other Methods. For many reinforcement learning algorithms, high-dimensional continuous state-spaces would be infeasible without manually designed features. For example, using generic features [Lagoudakis and Parr, 2003] yields impracticable feature dimensions on problems with 100 or more dimensions. For example, for a 100-dimensional state-space there would be 100^2 second-degree polynomials, or 2^{100} radial basis functions on the smallest possible grid. All comparative methods that performed reasonably rely on such a grid, so we will only show the results of the proposed method. For comparison, we show the results of the proposed

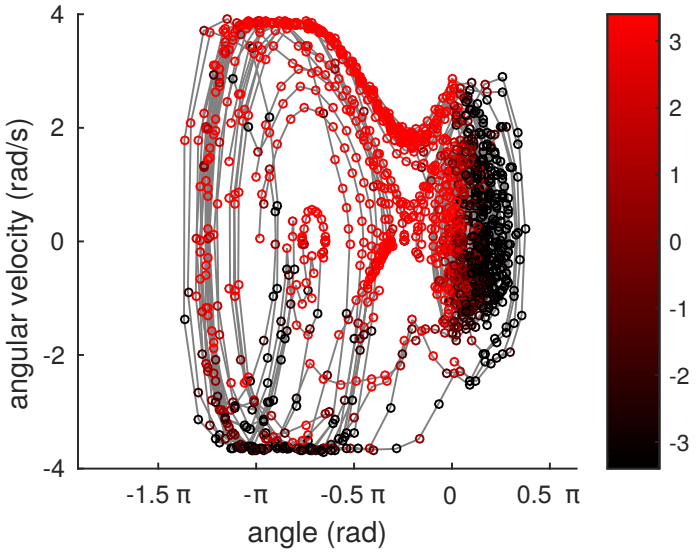


Figure 3.7.: Results of the real-robot experiment with high-dimensional state representations. Phase-space trajectories with torque. of one of the learned policies. Color indicates the applied torque (Nm). In most trajectories, the robot manages to swing up the pendulum and balance it around the upright position ($\theta = 0$). Near this position, the pendulum tends to oscillate as a consequence of time discretization and system delays.

method on an easier version of the task where the robot has access to the joint angle and angular velocity instead of the image-based representation.

Results for the Real-robot Swing-up with High-dimensional State Representations. The results of swing-up tasks with visual state features are shown in Figure 3.8. Of the six trials we performed, five resulted in policies that successfully swing-up and balance the pendulum. An example of the phase-space is shown in Figure 3.7. It is apparent that the pendulum oscillates near the target position which occasionally causes the pendulum to drop. The pendulum-camera system has some system delay which could cause such oscillations, and which also prevents a finer time discretization.

Figure 3.8 shows the average learning progress for the task. The system learns from an initial uninformed policy that hardly obtains high rewards, with the fastest learning progress obtained between 40 and 100 roll-outs. Our result shows that non-parametric methods are a promising approach to dealing with high-dimensional state representations such as images, since the task is successfully learned. Nevertheless, knowing a good representation, such as joint values, still resulted in better learning performance. Generally, many different representations could be used as long as a kernel can be defined that yields appropriate similarity values: in the end, the algorithm only performs operations based on those similarity values. Some pixels will never change their value. Such pixels are not problematic, as stationary kernels such as the squared exponential are not influenced by them.

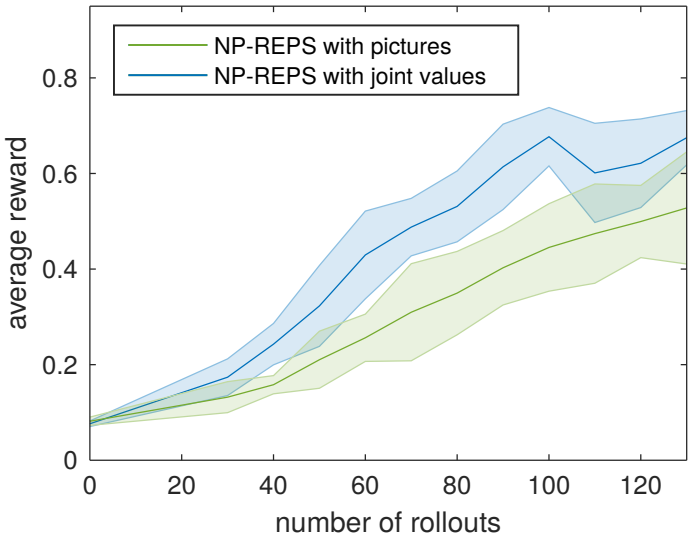


Figure 3.8.: Results of the real-robot experiment with high-dimensional state representations. Learning progress over six trials with input from either joint angles or pictures only. The REPS algorithm would converge to a locally optimum solution regardless of input modality, but seems to need more training time to reach such a local optimum using visual input. The graph shows the average reward over six independent trials. Error bars show twice the standard error.

3.3.7 Real-robot Tactile Control Experiment

In this last experiment, we aim to learn a control policy based on high-dimensional tactile signals, that are influenced by low-intensity noise. Although noisy observations make the system partially observable, this experiment will show us how robust the policy learning algorithm is to small violations of the Markov assumption. We consider learning policies based on either raw input signals or representations learned using the techniques described in Section 3.2.8.

The task is for the robot to manipulate a platform that it is touching with a Syntouch BioTac fingertip sensor. The platform can be rotated in roll (ψ_1) and pitch (ψ_2) directions. The robot itself has 5 degrees of freedom, as shown in Figure 3.9, and only observes the system state through its sensor. The sensor provides twelve readings of the 19 electrodes in each time step, resulting in 228-dimensional observations. Actions consist of an increment in task-space position, with the height of the robot computed to keep the pressure at the fingertip constant. The control frequency of the robot is 2.8 kHz, but desired task-space positions are kept constant during 33 ms time windows.

For the policy, we use the Fourier-transform based approximation discussed previously, since a policy with a fixed amount of parameters is easier to deploy on the robot system. The auto-encoder is configured with a hidden layer with 512 neurons and a feature layer with three neurons. These values were manually set based on the reconstruction error. To use the collected data efficiently, sampled data from all previous iterations was used to train the auto-encoder after every iteration. Since recent data

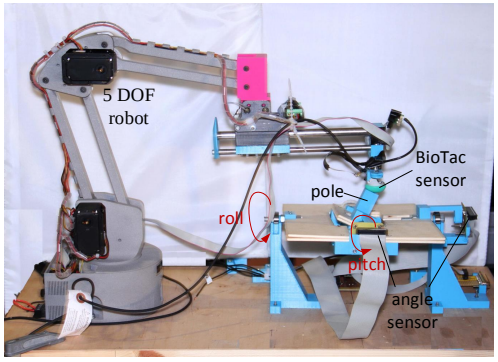


Figure 3.9.: Setup of the tactile control experiment. The five-DoF robot manipulates a platform that can rotate about two axes. The pitch and roll of the platform are measured to provide a feedback signal, but are not used in the control policy.

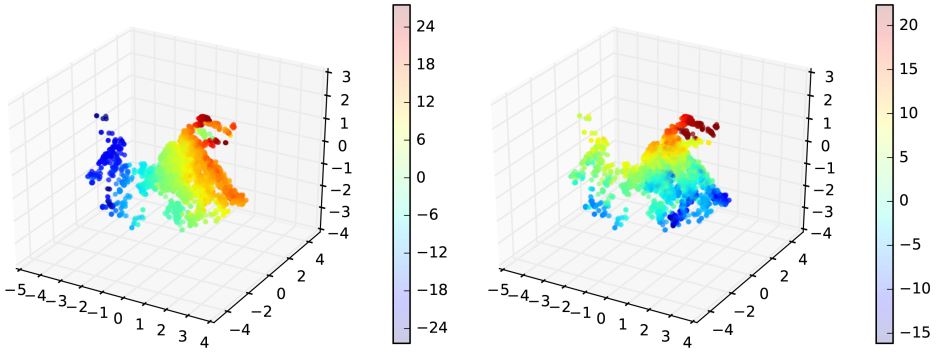


Figure 3.10.: Latent space for BioTac sensor. The x, y and z axis represent the latent values, the samples are colored according to roll and pitch of the platform. The visualization shows that in the learned feature space, the pitch and roll components are perpendicular to each other.

is more relevant, data from the most recent iteration was given triple weight and data from the previous iterations was given double weight in the objective function.

The goal of the robot to bring the platform to the equilibrium position is encoded by the reward function $\mathcal{R}^{\mathbf{a}}_{\mathbf{s}} = 60(\exp(-(\psi_1^2 + \psi_2^2)/60) - 1)$. 5 trials were performed of 15 iterations each. In each iteration, 15 roll-outs were performed (with 45 roll-outs used in the first iteration). The reset probability is set to 0.05, resulting in an average roll-out length of 20 time steps.

Results of the Tactile Control Experiment. The representation learned using the modified variational auto-encoder is shown in Figure 3.10. This figure shows that the learned representations smoothly represents high-dimensional sensor data in a manner that is consistent with the underlying roll and pitch angles (that were not available to the learner). The results of reinforcement learning of the tactile manipulation task with or without the encoder are shown in Figure 3.11. With the learned representation, RL progress is smooth and stable. The final policy brings the pole within 1° of the desired location in roll-outs with at least 10 time steps, on average. The performance

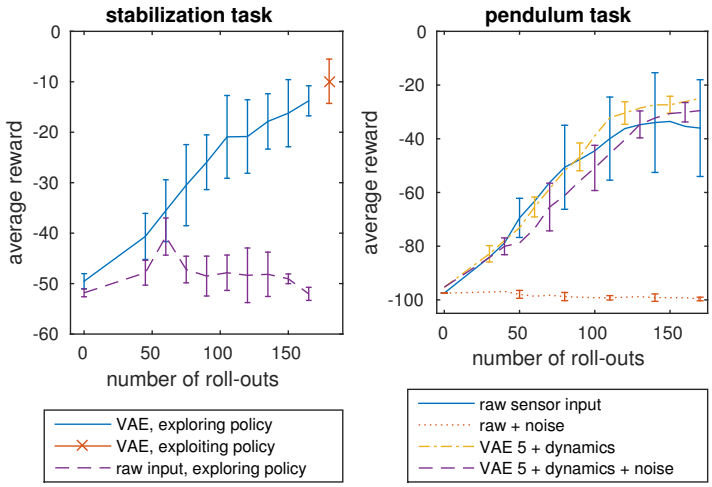


Figure 3.11.: Left: Learning progress on the real robot tactile manipulation task. During learning, performance of the learned policy including the learned exploration term is shown. After learning, we evaluate the learned mean policy without exploration. Error bars show the sample standard deviation. Averages are calculated over five independent runs, with independently learned feature encoders. Roll-outs contained 20 steps, on average. Right: On raw sensor data, the learner is very sensitive to noise. In the simulated experiment, performance collapses for noisy raw sensor inputs, but not when the learned representation is used.

of a baseline policy on the raw data directly is very poor. A possible cause for this poor performance is that the sensor data is too noisy.

To better understand the behavior of the system in the presence of observation noise, we evaluated the performance of reinforcement learners on the simulated visual pendulum swing-up described by van Hoof et al. [2015b] with low-intensity per-pixel noise (about 1% of the maximum image value). Figure 3.11 shows the performance on the noisy pictures, compared to the performance on the original task. When the images are corrupted with a small amount of noise, the learner using raw images indeed seems unable to learn the task.

3.4 Related Work

Over the years, many different reinforcement learning (RL) algorithms have been proposed, as reviewed by, among others, Bertsekas and Tsitsiklis [1996], Szepesvári [2010], Sutton and Barto [1998], Busoniu et al. [2010], Powell [2007], Bartlett [2003], Kaelbling et al. [1996], Wiering and Otterlo [2012] and Deisenroth et al. [2013]. Many of the most well-known algorithms are value-function methods. However, there is no notion of the sampled data or sampling policy in the value function, making it impossible to limit the loss of information as the policy is updated [Peters et al., 2010]. Furthermore, in continuous state-action spaces, such methods need to be approximated, which means that policy iteration does not necessarily improve the policy [Kakade and Langford, 2002, Bartlett, 2003].

Policy search methods are a complementary class of reinforcement learning approaches, which explicitly represent the sampling policy [Deisenroth et al., 2013]. Such methods might additionally represent a value function, e.g. [Williams, 1992, Peters and Schaal, 2008b, Sutton and Barto, 1998], in which case they are referred to as actor-critic algorithms. Policy search methods allow the learning agent to take the sampled data or the sampling policy into account. Other advantages include that policies might be easier to represent than value functions and convergent algorithms for policy search are known [Bagnell and Schneider, 2003a]. Furthermore, for certain policy parameterizations, stability and robustness guarantees can be given [Bertsekas, 1995]. If prior knowledge or task demonstrations are available, these can usually be integrated straightforwardly in policy search methods [Deisenroth et al., 2013, Peters and Schaal, 2006]. For these reasons, policy search method have in practice proven to work well on real (e.g. robotic) systems [Deisenroth et al., 2013].

In high-dimensional domains, both value-function and policy search methods have often relied on hand-crafted feature representations [Kaelbling et al., 1996, Kober et al., 2013, Bartlett, 2003]. This hand-tuning can largely be avoided by learning a suitable representation [Jonschkowski and Brock, 2015, Böhmer et al., 2013]. Such an approach, however, often requires a lot of training data and usually relies on non-convex optimization. Defining a representation can also be side-stepped by using non-parametric methods [Ormoneit and Sen, 2002, Rasmussen and Kuss, 2003], that implicitly use very rich representations that can adapt to the complexity of the data. Such methods have the disadvantage that they usually rely on inverting matrices that grow with the dataset, however. As a solution, efficient approximations can be used [Seeger et al., 2003, Snelson and Ghahramani, 2006, Rahimi and Recht, 2007].

We will discuss these issues in more detail in the remainder of this section. First, we will discuss various RL methods that provide stable policy updates. Subsequently, we will give an overview of different studies addressing RL with high-dimensional states, followed by a discussion of non-parametric RL techniques. Finally, we discuss various methods for efficiently approximating non-parametric methods that allow such methods to be applied to large data sets.

3.4.1 Policy Updates with Limited Information Loss.

Perhaps the most straightforward way to limit the information loss of a policy is to stay close to a previous policy. Policy gradient methods [Williams, 1992, Sutton et al., 1999a], for example, limit the policy update $\delta\boldsymbol{\theta}$ to a step in the gradient direction of fixed Euclidean length $\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta} = \epsilon$. However, this metric is not invariant to re-parametrization of the policy. This problem can be addressed by instead using the Fisher information metric $\mathbf{F}$ in the constraint $\delta\boldsymbol{\theta}^T\mathbf{F}(\boldsymbol{\theta})\delta\boldsymbol{\theta} = \varepsilon$, as suggested by Kakade [2002]. This constraint can be interpreted as the second-order Taylor expansion of the loss of information (relative entropy) between the path distributions of the original and the updated policy [Bagnell and Schneider, 2003a].

There are two main approaches to specifying such an information constraint. As suggested by Bagnell and Schneider [2003a], the loss of information between successive path distributions might be bounded. Such a formulation is equivalent to

bounding the expected loss of information between successive policies, since the transition dynamics are the same under both path distributions. Previous work [Peters et al., 2010], instead proposed to bound the information loss between successive state-action distributions.

The first formulation has led to various algorithms that provide stable policy updates. For example, the dynamic policy programming algorithm [Azar et al., 2011] uses the relative entropy between successive policies as an additional cost term in the value function. Similar update equation have been derived from two points of view. Firstly, from the point of view of maximizing the cumulative reward under constraints on the communication bandwidth or data processing capacity [Tishby and Polani, 2011], or, equivalently, minimizing the policy complexity under a constraint on the policies value [Still and Precup, 2012]. Another derivation minimizes the relative entropy from the trajectory distribution conditioned on obtaining maximal rewards to the proposed policy [Rawlik et al., 2013b]. In these approaches, the trade-off between greedy exploitation and maintaining stable updates—analogous to the (inverse) temperature of a Boltzmann distribution—is a free parameter. To obtain convergence, an update schedule that decays the temperature over time has to be designed. In contrast to adding the relative entropy as a cost-term, Levine and Abbeel [2014] employ a bound on the relative entropy between successive policies.

Another line of work, that is related to bounding the divergence between successive policies, has focused on guaranteeing improvement of the policy by limiting the policy update. In conservative policy iteration [Kakade and Langford, 2002] and save policy iteration [Pirotta et al., 2013], the updated policy is a mixture of the old policy and a greedy policy that maximized a lower bound on the policy improvement. Trust region policy optimization [Schulman et al., 2015] similarly guarantees policy improvement, using the relative entropy between successive policies in the lower bound.

Kober et al. [2013] observed that "in practice, reinforcement learning algorithms tend to work best for real systems when they are constrained to make modest changes to the distribution over states while learning". However, depending on the system dynamics, a small change in the policy might cause a large change in the distribution of states visited by the policy. Thus, it may be advantageous to limit the relative entropy between successive state-action distributions rather than between successive policies or trajectory distributions. Empirically, a bound on the state-action distribution has been shown to outperform a bound on the policy for relatively large step-sizes [Lioutikov et al., 2014]. Moreover, bounding the information loss between subsequent state-action distributions has been shown to have optimal regret in an adversarial Markov decision process (MDP) settings for discrete finite horizon problems [Zimin and Neu, 2013].

In previous work [Peters et al., 2010], the Relative Entropy Policy Search (REPS) algorithm was derived based on such a bound on the relative entropy between successive state-action distributions. Policies learned using REPS re-weight the previous state-action distribution using a soft-max of the advantage function. This soft-max policy is reminiscent of the common ad-hoc exploration-exploitation trade-off using Boltzmann exploration [Kaelbling et al., 1996, Sutton, 1990, Lin, 1993], expectation-maximization based updates [Peters and Schaal, 2007, Kober et al., 2011], and the previously discussed algorithms by Azar et al. [2011] and Rawlik et al. [2013b]. The

advantage of REPS is, that the 'temperature' parameter that governs exploration is set directly by the algorithm. As a result, the algorithm is invariant to re-scaling of the reward function, and the temperature automatically decreases as the policy converges.

Alternative techniques search an optimal policy within the space of policies of similar exponential form [Lever and Stafford, 2015, Bagnell and Schneider, 2003b]. In REPS, the exponential form does not result from an imposed search-space of the policy, but is a direct consequence of optimizing the bounded optimization problem. Under specific assumptions on the environment and the reward functions, exponential transformations of the value functions are also used in the definition of the optimal policy in the non-parametric method by Rawlik et al. [2013a]. The embedding of the exponentiated value function, however, suffers from numerical problems where the value function is low.

Earlier approaches based on the REPS formulation [Lioutikov et al., 2014, Kupcsik et al., 2013, Peters et al., 2010, Daniel et al., 2016a] have shown to be successful on a variety of problems. In all these approaches, a function-valued Lagrangian multiplier V emerges from the resulting optimization problem, which can be seen as a value function [Peters et al., 2010]. However, these approaches assume the Lagrangian multiplier V is linear in manually-defined features, making it hard to apply the algorithm to domains with high-dimensional sensor representations. We will relax this assumption, by requiring V to be a member of a non-linear RKHS, allowing implicit infinite feature representations. In contrast to earlier approaches based on the REPS optimization problem, our method can naturally handle non-parametric policies, such as Gaussian process policies.

REPS requires the estimation of the Bellman error. This estimation can for example be performed using a learned transition model. Thus far, work on learned transition models for REPS has been limited. The transition dynamics have been approximated by deterministic single-sample outcomes [Daniel et al., 2016a, Peters et al., 2010] which only works well for deterministic environments, or by time-dependent linear models [Lioutikov et al., 2014]. Gaussian process models have been used in the bandit setting [Kupcsik et al., 2013] to learn a simulator that predicts the outcome of new roll-outs. Instead, similar to previous value function methods [Grünewälder et al., 2012b, Boots et al., 2013, Nishiyama et al., 2012], we will employ empirical conditional RKHS embeddings to obtain non-linear models for step-based reinforcement learning.

3.4.2 Reinforcement Learning for High-dimensional State Representations.

Recently, several researchers have started to address the problem of reinforcement learning with high-dimensional state representations such as camera images. In many of these works, learning consists of two fully separate steps: learning a representation and, subsequently, learning a policy or value function. One possible approach is to view the problem as reducing the dimensionality of the high-dimensional states while losing as little information as possible. For such a purpose, deep neural networks such as deep auto-encoders have become popular. For example, Lange et al. [2012] and Mattner et al. [2012] used such networks to learn state representations that were subsequently exploited to learn visual tasks using fitted Q-iteration or non-parametric

approximate dynamic programming, respectively. Finn et al. [2016] used spatial auto-encoders, that exploit the spatial coherence of images, to learn visual pushing tasks using guided policy search.

The discussed methods do not explicitly make use of the structure of reinforcement learning problems. In domains where salient sensory distractors occur, taking such structure into account can help avoid representing those distractors in the learned representation. Jonschkowski and Brock [2015] proposed a method that takes observed transitions and rewards into account using a set of robotic priors, to learn representations for, among others, a visual navigation task. Another way to take the dynamics into account is to use slow-feature analysis, which encodes diffusion distances based on the transition kernel [Böhmer et al., 2013]. The learned representations were used in an LSPI approach to learn visual navigation. A visual navigation task was also addressed by Boots et al. [2011], who select features based on their ability to predict characteristics of possible future events. Forcing the dynamics to be linear in the latent space is another way of enforcing a task-relevant representation. Watter et al. [2015] showed this principle to be successful at learning representations for model-predictive control of several dynamical systems in an off-policy setting.

The goal of finding a state representation that takes the task structure into account can also be reached by directly learning neural network policies that map from high-dimensional sensory information to control actions. In such approaches, the hidden neurons act as feature representation. As the network is trained to reproduce correct actions, those neurons are optimized to provide a representation that helps the learning agent succeed at its task. For example, Koutník et al. [2013] used an evolutionary algorithm to search for network coefficients for a visual racing task in a compressed space, Lillicrap et al. [2015] learned neural-network policies using an actor-critic algorithm for learning dynamic tasks such as locomotion from pixel images, and Schulman et al. [2015] used trust-region policy optimization to optimize such policies to learn to play Atari games from pixel images as well as challenging locomotion tasks.

However, these methods tend to require on the order of a million sampled time steps, which might not be easy to obtain in a real-robot setting. For such real-robot tasks, Levine et al. [2016] proposed using optimized trajectories in a low-dimensional space to train a neural network policy that directly uses raw image data as input. Convenient low-dimensional spaces are, however, not always provided.

Another possibility is to use a neural network as a forward model for the dynamics and possibly the reward function. Optimal control methods can then be used to obtain optimal action. Recently, this approach was used to find optimal policies when only simulated images of the system are given [Assael et al., 2015, Wahlström et al., 2015]. The disadvantage of such methods is that gradients have to be propagated through a number of connections that is the product of the planning horizon and the network depth.

Mnih et al. [2015] instead directly used a neural network as an approximate Q function, as such an approach is able to scale to large networks and datasets. Their method obtained superb performance on playing Atari games but required training on millions of data-points. Such large datasets are impractical to obtain for physical systems, e.g., robots. Furthermore, all neural-network based learning methods rely on non-convex optimization of all the network weights.

3.4.3 Non-parametric Reinforcement Learning Methods.

Non-parametric kernel methods use an implicit representation of the data, and therefore avoid the explicit choice of a feature representation. Many different approaches in reinforcement learning have been re-formulated to use non-parametric methods. For example, non-parametric value iteration methods have been proposed for (partially observable) MDPs [Grünewälder et al., 2012b, Nishiyama et al., 2012], and non-parametric approximate dynamic programming method have been proposed by Ormoneit and Sen [2002], Taylor and Parr [2009], Deisenroth et al. [2009], Kroemer and Peters [2011] and Xu et al. [2014]. Engel et al. [2003] proposed a Gaussian process (GP) temporal difference algorithm, and furthermore, there are examples of non-parametric policy iteration schemes such as kernelized least-squares policy evaluation [Jung and Polani, 2007] and least-squares policy iteration [Xu et al., 2007, Rasmussen and Kuss, 2003]. The approximate linear programming algorithm, proposed by Pazis and Parr [2011], does not use kernels. Instead, this model-free method assumes the value function is Lipschitz, and assumes deterministic dynamics. Such value function methods use greedy maximization with respect to approximated value functions. Consequentially, these methods use deterministic actions. If exploration of the state space is required, heuristics such as an ϵ-greedy or a soft-max policy can be used. Furthermore, Grünewälder et al. [2012b] and Nishiyama et al. [2012] consider only discrete action sets and assume the state-action space can be sampled uniformly. For robotic systems, this can generally only be done in simulation.

Policy-search methods can be applied to address these shortcomings, by iteratively improving a policy. Kober et al. [2011] introduced cost-regularized kernel regression, which finds non-parametric policies for contextual bandits. In contrast, other approaches have focused on step-based decision making. For example, Bagnell and Schneider [2003b] developed a policy gradient method embedding a desirability function that defines a policy in a RKHS. However, their approach is restricted to discrete actions, and as a model-free method, cannot exploit learnable system dynamics. More recently, Lever and Stafford [2015] introduced another policy gradient approach, which searches for the mean function of a Gaussian process policy within a RKHS. Although this method cannot represent non-Gaussian policies unlike the method of Bagnell and Schneider [2003b], it can represent policies that are close to deterministic more easily. A disadvantage of this method is that a schedule to decay the covariance of the Gaussian towards zero has to be manually defined. Vien et al. [2016] introduced a non-parametric variant of the *natural* policy gradient, together with a natural actor-critic algorithm and expectation-maximization based updates.

Alternative non-parametric methods were proposed by Rawlik et al. [2013a] and Deisenroth and Rasmussen [2011]. The method by Rawlik et al. [2013a] considers continuous-time systems with continuous actions. This method assumes the environments injects observable control noise and that the system is control-affine. Deisenroth and Rasmussen [2011] describe a model-based iterative method. They explicitly marginalize the uncertain non-parametric model to avoid over-greedy optimization. However, their method requires the reward function to be known and to be of squared

exponential form. Additionally, it selects action greedily and so it does not address the exploration problem.

3.4.4 Efficient Approximation for Non-parametric RL Methods.

Kernel-based non-parametric methods usually requires inverting Gram matrices, which are of dimension $n \times n$, where n is the number of data points. This step is a bottleneck for scaling up these methods to large data sets (1000 - 10000 samples being the upper limit where most of these methods start to be prohibitively slow). Therefore, non-parametric approaches benefit from efficient approximations for large datasets.

Multiple non-parametric RL algorithms handle large datasets by selecting a subset of data points, and then finding coefficients for each of the kernels using a quadratic programming problem [Engel et al., 2003, Xu et al., 2014, Jung and Polani, 2007, Xu et al., 2007, Lever and Stafford, 2015]. In most cases, this sparsification approach results in an approximation of the kernel function by a non-stationary kernel function parametrized by a subset of active data-points, as pointed out by Jung and Polani [2007] and Xu et al. [2007]. If all data-points were chosen to be active, this approximation would be exact.

A sparsification approach for the method of Ormoneit and Sen [2002] has been proposed by Barreto et al. [2011]. The proposed stochastic factorization approach, however, works only on stochastic kernel matrices, such as used in Nadaraya-Watson kernel regression. Another approach for applying cost-regularized kernel regression to large data sets is to apply a learning algorithm to separate subsets of the data and combine the results [Macedo et al., 2014]. However, as the authors state, this approximation is not mathematically equivalent to the original problem, and if, as suggested, data from multiple iterations is combined, the on-policy assumption of the algorithm would be violated.

Yet another method takes an approach complimentary to sparsification approaches. Whereas sparse Gaussian process approaches essentially find coefficients for the true kernel matrix at a subset of data points, Rahimi and Recht [2007] propose an approach that evaluates an approximation to the kernel function at all training data points. To the best of our knowledge, this type of approximations has, so far, not been explored in the context of reinforcement learning. This approach approximates the kernel function as an inner product of random Fourier features. As such, it is reminiscent of the work by Fard et al. [2013] and Ghavamzadeh et al. [2010], who use random projections for parametric value function methods. However, these studies aim at reducing the dimensionality of sparse or redundant features using random projections, the method of Rahimi and Recht [2007] is used to project low-dimensional vectors into high-dimensional feature spaces so that the function approximation problems become linear. Konidaris et al. [2011] employ Fourier basis features for value function approximation. The parameters of the basis features were predetermined whereas in our approach the parameters are drawn from a distribution based on the kernel bandwidth, which can be optimized using standard techniques.

3.5 Conclusion

This section will first provide a summary of the contributions presenting in this chapter, as well as the results from experimental evaluation. After that, in the epilogue, we describe potential directions for future work.

3.5.1 Summary of this Chapter

In this chapter, we have developed a policy search method with smooth, robust updates to solve continuous MDPs. Our method uses learned non-parametric models and allows the use of non-parametric policies, avoiding hand-crafted features. By taking the sampling distribution into account during policy updates, stable learning progress was obtained even with relatively small batches of data. By embedding the conditional transition distribution, expectations over functions of the next state can be computed without density estimation. The resulting predictions are robust even in high-dimensional state spaces.

We show that the resulting algorithm is able to outperform other non-parametric algorithms on a reaching task and a pendulum swing-up task with control noise in on-policy settings. On-policy learning avoids the need to sample arbitrary state-action pairs.

A limiting factor in applying the method to larger problems is the computational cost of inverting the Gram matrix. To address this issue, we evaluated different approaches to approximate this matrix that allow the inverse to be calculated efficiently. We found the random Fourier features, which have not been used for reinforcement learning to date, to have desirable properties: they are computationally fast, easy to implement, and yielded good performance for moderate numbers of basis features. Sparsification yielded better performance on a smaller set of basis functions.

We evaluated the applicability of the algorithm to a real-robot underpowered swing-up task, with 600-dimensional visual representations of the pendulum's state. Here, we found that our method could successfully learn policies that swing up and balance the pendulum, from high-dimensional data.

On a real-robot tactile stabilization task, we showed that the robot was able to learn a policy that manipulates and stabilizes a platform. Rather then using joint encoders or handcrafted features, our algorithm learned the tasks based on features learned using a modified variational auto-encoder that represents complex and high dimensional tactile representations. Distances in the learned feature space are distorted less by noise on the input signal than distances in the raw input space. This explains why learning is only successful when the learned feature space is used when high-dimensional signals are noisy.

Many tasks concerning sensory data are similar to the real-robot underpowered swing-up task, in that they have a high extrinsic dimensionality, but are intrinsically low-dimensional. Kernel-based algorithms perform all operations on kernel values, so they are invariant to the extrinsic dimensionality. Our kernel-based RL algorithm can, thus, be applied to such tasks without the explicit dimension reduction step used by many existing methods.

3.5.2 Epilogue

As shown in this chapter, non-parametric relative entropy policy search seems a promising reinforcement learning method, especially in real-robot problems with high-dimensional sensory data and limited interaction time. Most sections of this chapter are from a paper currently under review with the Journal of Machine Learning Research [van Hoof et al., 2016b]. This paper is itself based on earlier conference papers [van Hoof et al., 2015b,a], as well as the foundational work by Peters et al. [2010]. Sections 3.2.7 and 3.3.7 have previously been published in the International Conference on Intelligent Robots and Systems [van Hoof et al., 2016a]. Sections 3.4.2 and 3.5.1 are based on both of these papers.

Certain aspects of the proposed method could be improved or extended in future work to improve learning performance or scale to larger domains. For example, the fitting of the generalizing policy is a step that takes up a significant proportion of the total computation time, but might fail to encode the desired distribution. In contextual bandit problems with an approximately quadratic reward function, this step can be avoided [Abdolmaleki et al., 2015]. A similar method could be applied for reinforcement learning with multi-step episodes, by fitting a quadratic function to the Bellman error δ. Although this procedure loses part of the information in the Bellman error, it prevents a potentially larger loss in the subsequent fitting of the policy. If a global quadratic approximation is not suitable, the Bellman error could also be approximated on-the-fly for each current state. Thereby, the quadratic function would only need to approximate the relationship between the control action and the Bellman error for a single state, at the cost of higher computational cost at runtime.

Our learned representations aimed at predicting the observation at the next time step. Although this procedure can help learn models more robustly [van Hoof et al., 2016a], it would still result in representations that include task-irrelevant distractors if they are consistent over time [Jonschkowski and Brock, 2015]. Finding a more principled approach that avoids representing distractors could yield more accurate representations. Optimization of the kernel hyper-parameters that specify the representation of the value function does not always yield desirable values, especially in systems with multi-dimensional controls. Thus, learning of representations stays an important issue for future work.

In relative entropy policy search, there is an intricate relationship between the Lagrangian multiplier V and the proposed state-action distribution p_π. As a result, in contrast to value function methods such as least-squared temporal difference learning (LSTD), there is no closed-form solution for the parameters of V. As an alternative, Wirth et al. [2015] maximizes the expected value of the Q function. Thus, a solution is obtained that is very similar to that of a contextual bandit, where efficient solutions exist [Abdolmaleki et al., 2015]. However, this strategy only limits the expected KL divergence between successive policies, but not between successive steady-state distributions. In fact, the state distribution is assumed constant. Another approach would be to exploit the form of the dual function, which is composed of additive terms. Like the approach by Levine et al. [2016], the alternating direction method of multipliers (ADMM) could help optimize this function faster.

Learning 'from scratch' is limited by the availability of sufficient training data. Impressive recent demonstrations of reinforcement learning techniques have tended to rely on the availability of massive amounts of data [Mnih et al., 2015] or by using additional information, such as task-space coordinates [Levine et al., 2016]. The amount of data a robot can collect is usually limited by hardware availability, wear and tear on the robot, or the need for a human in the loop. Therefore, of these two approaches, learning with more prior information seems a more suitable approach for robot learning. Often, such prior knowledge is already available, for example, in the form of a kinematic model. Integrating different forms of prior knowledge, but still allowing aspects of the robot or environment that are unknown or inaccurately modeled to be learned is a challenging topic for future work.

In our proposed algorithm, like most other reinforcement learning algorithms, exploration is performed by perturbing actions at individual time-steps independently. This approach can yield inefficient random walk behavior and can make learning more fragile in real robot systems. In the next chapter, we will discuss these issues in more detail and propose a possible solution.

3.A The Dual and its Derivatives

To obtain the dual function, we re-insert the state-action probabilities $p_\pi = \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ in the Lagrangian to obtain the dual

$$\begin{aligned}
g(\eta, V, \lambda) =& \lambda + \eta\epsilon + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}\left[\delta(\mathbf{s},\mathbf{a},V) - \lambda - \eta\log\frac{p_\pi(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})}\right], \\
=& \lambda + \eta\epsilon + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}[-\lambda + \lambda] + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}[\delta(\mathbf{s},\mathbf{a},V) - \delta(\mathbf{s},\mathbf{a},V) + \eta], \\
=& \lambda + \eta\epsilon + \mathbb{E}_{p_\pi(\mathbf{s},\mathbf{a})}\eta \; \mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} = \lambda + \eta\epsilon + \eta = \eta\epsilon + \eta\log(Z), \\
\approx& \eta\epsilon + \eta\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp\left(\delta(\mathbf{s}_i,\mathbf{a}_i,V)/\eta\right)\right),
\end{aligned}$$

where we used that $\exp(-\lambda/\eta - 1) = Z^{-1}$, so $\lambda + \eta = \eta\log(Z)$. In the last line, the expected value over q is approximated by taking the average of samples $1, \ldots, n$ taken from q. Note that λ and q do not appear in the final expression.

When employing the kernel embedding, the Bellman error is written as $\delta(\mathbf{s}_i, \mathbf{a}_i, \boldsymbol{\alpha}) = \mathcal{R}_{\mathbf{s}_i}^{\mathbf{a}_i} + \boldsymbol{\alpha}^T(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i))$. We define

$$w_i = \frac{\exp\left(\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})/\eta\right)}{\sum_{i=j}^{n}\exp\left(\delta(\mathbf{s}_j,\mathbf{a}_j,\boldsymbol{\alpha})/\eta\right)}$$

to keep equations brief and readable. The partial derivatives can be written as:

$$\begin{aligned}
\frac{\partial g(\eta,\boldsymbol{\alpha})}{\partial\eta} =& -\frac{1}{\eta}\sum_{i=1}^{n} w_i\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha}) + \epsilon + \log\left(\frac{1}{n}\sum_{i=1}^{n}\exp\left(\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})/\eta\right)\right), \\
\frac{\partial g(\eta,\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}} =& \sum_{i=1}^{n} w_i\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right),
\end{aligned}$$

and furthermore, we obtain the partial Hessians:

$$\begin{aligned}
\frac{\partial^2 g(\eta,\boldsymbol{\alpha})}{\partial\eta\partial\eta} =& \frac{1}{\eta}\sum_{i=1}^{n} w_i\left(\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})\right)^2 - \frac{1}{\eta}\left(\sum_{i=1}^{n} w_i\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})\right)^2, \\
\frac{\partial^2 g(\eta,\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\alpha}^T} =& -\frac{1}{\eta}\left(\sum_{i=1}^{n} w_i\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right)\right)\left(\sum_{i=1}^{n} w_i\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right)^T\right) \\
&+ \sum_{i=1}^{n}\frac{w_i}{\eta}\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right)\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right)^T, \\
\frac{\partial^2 g(\eta,\boldsymbol{\alpha})}{\partial\eta\partial\boldsymbol{\alpha}} =& \left(\sum_{i=1}^{n}\frac{w_i}{\eta}\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})\right)\left(\sum_{i=1}^{n} w_i\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right)\right) \\
&- \sum_{i=1}^{n}\frac{w_i}{\eta}\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})\left(\mathbf{K}\beta(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{k}_\mathrm{s}(\mathbf{s}_i)\right).
\end{aligned}$$

3.B Optimization with Respect to V

We want to show that at least one of the functions V minimizing the dual functional g can be represented using a weighted sum of kernel functions centered at the sample states, i.e., that

$$V^* = \sum_{\tilde{\mathbf{s}} \in \tilde{\mathcal{S}}} \alpha_{\tilde{\mathbf{s}}} k_s(\tilde{\mathbf{s}}, \cdot), \tag{3.18}$$

where $\tilde{\mathcal{S}}$ is the set of states samples during the roll-outs. We follow some steps in the proof of Schölkopf et al. (2001). They consider arbitrary objective functions c mapping to $\mathbb{R} \cup \{\infty\}$ of the form

$$c((\mathbf{s}_1, y_1, V(\mathbf{s}_1)), \ldots, (\mathbf{s}_m, y_m, V(\mathbf{s}_m))), \tag{3.19}$$

which can represent an error function between the value of a function $V(\mathbf{s})$ at the samples $\mathbf{s}_i$ and the corresponding desired outputs y_i. In our case, we do not have desired output values y_i for our objective function. This is inconsequential as c can be arbitrary, and so can be independent of all y values.

Any function V can be written as $V = \sum_{\tilde{\mathbf{s}} \in \tilde{\mathcal{S}}} \alpha_{\tilde{\mathbf{s}}} k_s(\tilde{\mathbf{s}}, \cdot) + v(\mathbf{s})$, where $v(\mathbf{s})$ is an additional bias term. If V is constrained to be in the Hilbert space defined by k, Schölkopf et al. [2001] show that c is independent of the bias term $v(\mathbf{s})$. This means that for any optimal V' that is not of the proposed form, there is a V^* of the proposed form that has the same objective value which is obtained by subtracting $v(\mathbf{s})$ from V'.

As the dual function g satisfies the conditions to cost function c, for us this means that there is at least one V^* optimizing g of the proposed form. Note that it is inconsequential that the dual g also depends on Lagrangian parameter η. For any optimum $(\eta^*, V^{*\prime})$, if $V^{*\prime}$ is not of the proposed form, projecting $V^{*\prime}$ on the proposed basis yields another function V^* that satisfies $g(\eta^*, V'^*) = g(\eta^*, V^*)$, so (η^*, V^*) must be an optimum as well.

Therefore, there is always at least one minimum of any such function c of the proposed form.

3.C Feedback Signals to Avoid Joint Angle and Velocity Limits

In our real robot setup, the control actions selected by the algorithm is the increment in the torque to be applied (proportional to the jerk). However, to avoid running into joint angle and velocity limits, the resulting torque is modified when these limits are approaches. Therefore, the actually applied torque at time-step t

$$\tau^{(t+1)} = \max\Big(\min\Big(\tau_t + u + \tau_{\mathrm{s}}^{(t)} + \tau_{\mathrm{d}}^{(t)}, \tau_{\max}^{(t)}\Big), \tau_{\min}^{(t)}\Big),$$

where $\tau_{\mathrm{s}}^{(t)}$ and $\tau_{\mathrm{s}}^{(d)}$ are spring- and damper related terms that apply when the robot gets close to the joint limit, and $\tau_{\max}^{(t)}$ and $\tau_{\min}^{(t)}$ are the maximum and minimum torque, respectively, that can be applied without breaking the velocity limit. The definition of θ, as well as the joint limit and the area where the feedback terms are applied, are illustrated in Figure 3.5b.

The feedback terms are defined as follows. The spring-like term

$$\tau_{\mathrm{s}}^{(t)} = \begin{cases} 0 & \text{if } -5/4\pi < \theta < 1/4, \pi \\ 15(-5/4\pi - \theta) & \text{if } \theta < -5/4\pi, \\ 15(1/4\pi - \theta) & \text{if } 1/4\pi < \theta, \end{cases}$$

is applied whenever the joint gets close to the joint limit at $1/2\pi = -3/2\pi$. The damper-like term

$$\tau_{\mathrm{d}}^{(t)} = \begin{cases} 0.3(-5/4\pi - \theta)\dot{\theta} & \text{if } \theta < -5/4\pi \text{ and } \dot{\theta} > 0, \\ 0.3(1/4\pi - \theta)\dot{\theta} & \text{if } 1/4\pi < \theta \text{ and } \dot{\theta} < 0, \\ 0 & \text{otherwise,} \end{cases}$$

is applied in the same region, as the feedback has to be higher when the pendulum has a high velocity. To prevent the velocity from exceeding the velocity limit, additionally, the minimum and maximum torques

$$\tau_{\max}^{(t)} = \min\big(\tau_{\max}^{(t-1)} + 0.05, 20(3.85 - \dot{\theta}) + 4.5\cos(\theta)\big),$$

$$\tau_{\min}^{(t)} = \max\Big(\tau_{\min}^{(t-1)} - 0.05, 20(-3.7 - \dot{\theta}) + 4.5\cos(\theta)\Big),$$

are applied. The cosine term is a rough compensation for the torque induced by gravity. The term linear in $\dot{\theta}$ is a linear damping term. Furthermore, the system has some delays which tended to induce oscillations close to the maximum torque. Therefore, the maximum (respectively minimum) can only be increased (decreased) by a small amount relative to the previous value.

4 Generalized Exploration in Policy Search

In the previous two chapters, we have described exploration for learning with an explicit task, and learning without an explicit task where the goal is to improve the understanding of the environment.

Exploration often has a random component, which typically yields independent perturbations of actions or paramters at each time step [Kaelbling et al., 1996]. Since opposite perturbations that cancel each other out might be applied at subsequent time steps, this behavior can be considered *incoherent* exploration. Such incoherent exploration has a couple of disadvantages: among others, high-frequency perturbations lead to 'washing out' of the exploration signals, high jerks are applied on the robot, and the system is sensitive to communication delays [Kober and Peters, 2009].

These problems can be addressed by perturbing the parameter at the beginning of the roll-out, and keeping this perturbation constant thoughout the roll-out [Rückstieß et al., 2010, Kober and Peters, 2009, Sehnke et al., 2010]. In this case exploration is completely *coherent*. Thus, instead of exploring locally around the average trajectory, different global strategies can be evaluated. Such an approach largely solves the issues mentioned in the previous paragraph. However, since only one perturbation can be evaluated per roll-out, this strategy could require more roll-outs, and thus require more interaction time on the robot, for each policy improvement step.

In this chapter, we investigate a generalized exploration strategy that has a *coherence trade-off* parameter, that can be set to reproduce completely incoherent exploration or completely coherent exploration, but that can also be set to intermediate values to yield behavior that combines some of the advantages of both extreme strategies.

4.1 Introduction

Obtaining optimal behavior from experience in unknown environments is formalized in the reinforcement learning (RL) framework [Sutton and Barto, 1998]. To learn in this manner, addressing the exploration/exploitation trade-off, that is, choosing between actions known to be good and actions that could prove to be better, is critical for improving skill performance in the long run. In fact, many reinforcement learning techniques require a non-zero probability of trying each action in every state to be able to proof that the algorithm convergences to the optimal policy [Sutton and Barto, 1998].

Most tasks require agents to make a sequence of decisions over multiple time steps. Typical algorithms perform exploration by modifying the action taken at some or all of the time steps. Popular exploration heuristics include ϵ-greedy action selection (choosing a random action in a fraction ϵ of time steps), use of a stochastic con-

troller that injects random noise at every time step, and by using a soft-max (or Boltzmann) distribution that selects actions that are deemed better more often, but not exclusively [Kaelbling et al., 1996, Deisenroth et al., 2013, Kober et al., 2013]. Another strategy is the use of parametrized controllers with a distribution over actions or parameters, and sampling from this distribution at every time step [Deisenroth et al., 2009].

However, the paradigm of modifying actions at individual time-steps has multiple shortcoming. High-frequency exploration can show inefficient 'thrashing' behavior [Strens, 2000, Osband et al., 2016, Asmuth et al., 2009] and in the worst case exhibit a random walk behavior that fails to explore much of the state space [Kober and Peters, 2009]. At the same time, for short time steps the variance of policy roll-outs explodes as the results depends on an increasing number of independent decisions [Munos, 2006]. Furthermore, when learning controllers within a certain function class, perturbing single time-steps can result in trajectories that are not reproducible by any noise-free controller in that function class [Deisenroth et al., 2013].

Skill learning in robotics and other physical systems is a prominent application domain for reinforcement learning. In this domain, reinforcement learning offers a strategy for acquiring skills when, for example, parts of the robot or parts of the environment cannot be modeled precisely in advance [Kaelbling et al., 1996, Kober et al., 2013]. High-frequency exploration can cause additional problems when applied on robot systems. Namely, high-frequency exploration causes high jerks, that can damage robots [Deisenroth et al., 2013, Kober and Peters, 2009, Wawrzyński, 2015]. Furthermore, real robots exhibit non-Markov effects such as dead-band, hysteresis, stiction, and delays due to processing and communication delays and inertia [Kober et al., 2013]. These effects make it hard to precisely measure the effects of the perturbations. Although these last effects could be addressed by including a history of actions to the state-space, these would make the dimensionality of the reinforcement learning problem harder and thereby increase the complexity exponentially [Kober et al., 2013].

In this paper, we focus on addressing these problems in policy search methods employing parametrized controllers. Such methods, that are popular in e.g. robotics applications, tend to yield stable updates that result in safe robot behavior [Kober et al., 2013, Deisenroth et al., 2013]. Parametrized policies are also easily applicable in environments with continuous state-action spaces. In these methods, perturbing individual actions can be realized by perturbing the policy parameters in each time step independently. We will refer to this strategy as *time-step-based exploration*.

The problems of high-frequency exploration in policy search methods can be addressed by exploiting that data for learning tasks through reinforcement learning is usually gathered in multiple episodes. One episode is a sequence of state-action pairs, that is ended when a terminal state is reached or a certain number of actions have been performed. One can thus perturb the controller parameters at the beginning of a policy roll-out, and leave it fixed until the episode has ended [van Hoof et al., 2015b].

The advantage of this *episode-based exploration* approach is that random-walk behavior and high jerks are avoided, and optimal controller parameters can be found even in the case of non-Markov effects. Since policy parameters stay constant during an episode, the resulting behavior is more *coherent* than the high-jerk random walk

behavior of time-step-based exploration strategies. The disadvantage, however, is that in each episode, only one set of parameters can be evaluated. Therefore, such techniques might require more episodes to be performed, which can be time-consuming on a robotic system.

We think of the time-step-based and episode-based exploration strategies as two extremes, with space for many different trade-offs in between them. In this paper, we *provide a unifying view* on time-step-based and episode-based exploration and *propose intermediate trade-offs* that slowly vary the controller parameters during an episode, rather than independent sampling or keeping the parameters constant. Formally, we will sample parameters at each time step in a manner that is dependent on the previous parameters, thereby defining a Markov chain in parameter space. Our experiments compare such intermediate trade-offs to prior step-based and episode-based methods.

In the rest of this section, we will describe related work, and then describe our unified view on time-step-based and episode-based exploration and our problem statement. Then, in the subsequent sections, we describe our approach formally and provide the details of the set-up and results of our experiments. We conclude with a discussion of the results and future work.

4.1.1 Related Work

Numerous prior studies have addressed the topic of temporal coherence in reinforcement learning, although most of these work have not considered finding trade-offs between fully temporally correlated and fully independent exploration. Different approaches have been proposed. In this section, we will first discuss temporal coherence through the use of options and macro-actions. Then, the possibility of temporal coherence through the use of parametrized controllers such as central pattern generators and movement primitives is discussed. Since we propose performing exploration by building up a Markov-chain in the parameter space, we will finally also discuss approaches that use sampling for generating exploratory actions or policies.

Temporal Coherence through Options

Hierarchical reinforcement learning has been proposed to scale reinforcement learning to larger domains, especially where common subtasks are important [Kaelbling, 1993, Singh, 1992]. These early studies allowed choosing higher-level actions at every time step, and are thus time-step base strategies. Later approaches tended to have a higher-level policy which select a lower-level policy that takes control for a number of time steps, for example, until the lower level policy reaches a specific state, or when a certain number of time steps has passed [Precup, 2000, Parr and Russell, 1998, Dietterich, 2000, Sutton et al., 1999b]. Choosing such a lower-level policy to be executed for multiple time steps makes the subsequent exploration decisions highly correlated. Moreover, this hierarchical framework allows learning to scale up to larger domains efficiently [Sutton et al., 1999b, Kaelbling, 1993, Parr and Russell, 1998, Dietterich, 2000]. In such hierarchical framework, the temporal coherence

of exploration behavior contributes to this success by requiring fewer correct subsequent decisions for reaching a desired, but faraway, part of the state space [Sutton et al., 1999b].

Much of this work has considered discrete Markov decision processes (MDPs), and does not naturally extend to robotic settings. Other has focused on continuous state-action spaces. For example, Morimoto and Doya [2001] study an upper level policy that sets sub-goals that provide a reward for lower-level policies. This method was used to learn a stand-up behavior for a three-link robot. A similar set-up was used in [Ghavamzadeh and Mahadevan, 2003], where the agent could choose between setting a sub-goal and executing a primitive action. Local policies are often easier to learn than global policies. This insight was used in [Konidaris and Barto, 2009] in an option discovery framework, where a chain of sub-policies is build so that each sub-policy terminates in an area where its successor can be initiated. Another option discovery method is described in [Daniel et al., 2016b], where probabilistic inference is used to find reward-maximizing options for, among others, a pendulum swing-up task.

Trajectory Based Exploration and Pattern Generators

The option framework is a powerful approach for temporally correlated exploration in hierarchical domains. However, option-based methods usually require the options to be pre-defined, require additional information such as the goal location, demonstrations, or knowledge of the transition dynamics, or are intrinsically linked to specific RL approaches. Another approach to obtaining coherent exploration employs parametrized controllers, where the parameters are fixed for an entire trajectory. Such an approach is commonly used with pattern generators such as motion primitives.

Such episode-based exploration has been advocated in a robotics context in previous work. For example, [Rückstieß et al., 2010, Sehnke et al., 2010] describe a policy gradient method that explores by sampling parameters in the beginning of exploration. This method is shown to outperform similar policy gradient methods which use independent Gaussian noise at each time step for exploration. One of the proposed reason for this effect, is that in policy gradient methods, the variance of gradient estimates increases linearly with the length of the history considered [Munos, 2006]. Similarly, the PoWER method that uses episode-based exploration [Kober and Peters, 2009] outperforms a baseline that uses independent additive noise at each time step. Furthermore, path-integral based methods have been shown to benefit from parameter-based exploration [Theodorou et al., 2010, Stulp and Sigaud, 2012], with episode-based exploration conjectured to produce more reliable updates [Stulp and Sigaud, 2012]. Rückstieß et al. [2010] propose to add a parametrized exploration policy on top of other policies such as value-function maximizing actions. The parameters of this exploration policy are similarly fixed throughout an episode, so that in the same state, the same action will consistently be chosen.

Episode-based exploration has been shown to have very good results where policies have a structure that fits the task. For example, in [Kohl and Stone, 2004], a task-specific parametrized policy was learned for quadrupedal locomotion using a policy gradient method. Dynamic movement primitives have proven to be a pop-

ular policy parametrization for a wide variety of robot skills [Schaal et al., 2005]. For example, reaching, ball-in-a-cup, under actuated swing-up and many other tasks have been learned in this manner [Kober and Peters, 2009, Schaal et al., 2005, Kober et al., 2013]. In case different initial situations require different controllers, a policy is can be found that maps initial state features to controller parameters [da Silva et al., 2012, Daniel et al., 2016a].

However, episode-based exploration also has disadvantages. Notably, in every roll-out only a single set of parameters can be evaluated. Compared to independent per-step exploration, many more roll-outs might need to be performed. Performing such roll-outs can be time-consuming and wear out the mechanisms of the robot. One solution would be to keep exploration fixed for a number of time steps, but then choose different exploration parameters. Such an approach was proposed in [Munos, 2006]. A similar effect can be reached by sequencing the execution of parametrized skills, as demonstrated in [Stulp and Schaal, 2011, Daniel et al., 2016a]. However, suddenly switching exploration parameters might again cause undesired high jerks in robot systems. Instead, slowly varying the exploration parameters is a promising strategy. Such a strategy is touched upon in [Deisenroth et al., 2013], but has remained largely unexplored so far.

Sampling for Reinforcement Learning

In this paper, we propose building a Markov chain in parameter space obtain coherent exploration behavior. Earlier work has used Markov chain Monte Carlo (MCMC) methods for reinforcement learning, but usually in a substantially different context. For example, several papers focus on sampling models or value functions. In case models are sampled, actions are typically generated by computing the optimal action with respect to the sampled model [Asmuth et al., 2009, Strens, 2000, Ortega and Braun, 2010, Dearden et al., 1999, Doshi-Velez et al., 2010]. By preserving the sampled model for multiple time steps or an entire roll-out, consistent exploration is obtained [Strens, 2000, Asmuth et al., 2009]. Such methods cannot be applied if the model class in unknown. Instead, samples can be generated from a distribution over value functions [Wyatt, 1998, Dearden et al., 1998, Osband et al., 2016]. Again, preserving the sample over an episode avoids dithering by making exploration consistent for multiple time-steps [Osband et al., 2016].

Instead, in this paper, we propose sampling policies from a learned distribution. Earlier work has used MCMC principles to build a chain of policies. This category of work includes [Hoffman et al., 2007] and [Kormushev and Caldwell, 2012], who use the estimated value of policies as re-weighting of the parameter distribution, [Wingate et al., 2011], where structured policies are learned so that experience in one state can shape the prior for other states, and [Watkins and Buttkewitz, 2014], where a parallel between such MCMC methods and genetic algorithms is explored. In those works, every policy is evaluated in an episode-based manner, whereas we want an algorithm that is able to explore during the course of an episode.

Such a method that explores during the course of an episode was considered in [Guo et al., 2004], which proposes a change to a single element of a tabular deterministic policy at every time-step. These proposals are accepted subject to the metropolis

criterion. However, this algorithm does not consider stochastic or continuous policies that are needed in continuous-state, continuous-action MDPs.

The work that is most closely related to our approach, is the use of auto-correlated Gaussian noise during exploration. This type of exploration was considered for learning robot tasks in [Wawrzyński, 2015] for learning robotic policies. In a similar manner, Ornstein-Uhlenbeck processes can be used to generate policy pertubations [Lillicrap et al., 2016, Hausknecht and Stone, 2016]. However, in contrast to the method we propose, these approaches perturb the actions themselves instead of the underlying parameters, and can therefore generate actions sequences that cannot be followed by the noise-free parametric policy.

4.1.2 Notation in Reinforcement Learning and Policy Search

Reinforcement-learning problems can generally be formalized as Markov decision processes. A Markov decision process is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, the probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a})$ that executing action $\mathbf{a}$ in state $\mathbf{s}_t$ will result in state $\mathbf{s}_{t+1}$ at the next time step, and a reward function $r(\mathbf{s}, \mathbf{a})$. In our work, we will investigate the efficacy of our methods in different dynamical systems. Thus, we will work with continuous state and action spaces, with $\mathbf{s}_t \in \mathcal{S} \subset \mathbb{R}^{D_s}$ and $\mathbf{a}_t \in \mathcal{A} \subset \mathbb{R}^{D_a}$, where D_s and D_a are the dimensionality of the state and action space, respectively. Also, the transition distribution $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a})$ is given by the physics of the system, and will thus generally be a delta distribution.

Our work focuses on policy search methods to find optimal controllers for such systems. In policy search methods, the policy is explicitly represented. Often, this policy is parametrized by a parameter vector $\boldsymbol{\theta}$. The policy can be deterministic or stochastic given these parameters. Deterministic policies will be denoted as a function $\mathbf{a} = \pi(\mathbf{s}; \boldsymbol{\theta})$, whereas stochastic policies will be denoted as a conditional distribution $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$.

4.1.3 Unifying View on Step- and Episode-based Exploration

In this paper, we will look at parameter-exploring policy search methods. Existing methods in this category have almost exclusively performed exploration by either performing exploration at the episode level or performing exploration at the step-based level. A unifying view on such methods is, that we have a (potentially temporally coherent) policy of the form

$$\mathbf{a}_t \sim \pi(\mathbf{s}_t; \boldsymbol{\theta}_t) \tag{4.1}$$

$$\boldsymbol{\theta}_t \sim \begin{cases} p_0(\cdot) & \text{if } t = 0 \\ p(\cdot|\boldsymbol{\theta}_{t-1}) & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\boldsymbol{\theta}_t$ is the vector of parameters at time t, $\mathbf{a}_t$ is the corresponding action taken in state $\mathbf{s}_t$, π is a policy that is deterministic given the parameters. The familiar step-based exploration algorithms correspond to the specific case where $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = p_0(\boldsymbol{\theta}_t)$, such that $\boldsymbol{\theta}_t \perp \boldsymbol{\theta}_{t-1}$. Episode-based exploration is another extreme case,

where $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = \delta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$, where δ is the Dirac delta, such that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$. Note, that in both cases

$$\forall t : \int p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0)p_0(\boldsymbol{\theta}_0)d\boldsymbol{\theta}_0 = p_0(\boldsymbol{\theta}). \tag{4.3}$$

That is, the marginal distribution is equal to the desired sampling distribution p_0 regardless of the time step. Besides these extreme choices of $p(\cdot|\boldsymbol{\theta}_{t-1})$, many other exploration schemes are conceivable. Specifically, in this paper we address choosing $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ such that the $\boldsymbol{\theta}_t$ is neither independent of nor equal to $\boldsymbol{\theta}_{t-1}$ and Eq. (4.3) is satisfied.

4.2 Generalizing Exploration

Equation (4.2) defines a Markov chain on the policy parameters. To satisfy Eq. (4.3), p_0 should be a stationary distribution of this chain. A sufficient condition for this property to hold, is that detailed balance is satisfied [Hastings, 1970]. In other words, the proposal distribution $g(\boldsymbol{\theta}|\boldsymbol{\theta}')$ that proposes a distribution for the next parameter value $\boldsymbol{\theta}$ given the current value $\boldsymbol{\theta}'$ and the acceptance probability $A(\boldsymbol{\theta}'|\boldsymbol{\theta})$ for such proposals should follow

$$\frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta}')} = \frac{A(\boldsymbol{\theta}|\boldsymbol{\theta}')g(\boldsymbol{\theta}|\boldsymbol{\theta}')}{A(\boldsymbol{\theta}'|\boldsymbol{\theta})g(\boldsymbol{\theta}'|\boldsymbol{\theta})}. \tag{4.4}$$

This constraint can be satisfied by choosing the acceptance probability as

$$A(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \min\left(1, \frac{p_0(\boldsymbol{\theta}')}{p_0(\boldsymbol{\theta})}\frac{g(\boldsymbol{\theta}|\boldsymbol{\theta}')}{g(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right).$$

In this paper, we will focus on Gaussian policies p_0. In this case, we will see that the proposal distribution can be chosen such that the acceptance probability $A(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is always 1.

Given a Gaussian policy[1] $p_0 = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$, a reasonable proposal distribution could be obtained by taking a weighted average of the parameters $\boldsymbol{\theta}_t$ at the current time step and a sample from a Gaussian centered on $\boldsymbol{\mu}$. Since averaging lowers the variance, this Gaussian will need to have a larger variance than $\boldsymbol{\Lambda}^{-1}$. As such, we consider a proposal distribution of the form

$$\boldsymbol{\theta}_{t+1} = \beta\tilde{\boldsymbol{\theta}} + (1-\beta)\boldsymbol{\theta}_t, \quad \tilde{\boldsymbol{\theta}} \sim N(\boldsymbol{\mu}, f(\beta)^2\boldsymbol{\Lambda}^{-1}), \tag{4.5}$$

where β is the weighting of the average and $f(\beta)$ governs the additional scaling of the covariance. This scaling needs to be set such that the detailed balance criterion in Eq. (4.4) is satisfied. The detailed balance criterion can most easily be verified by

[1] Such Gaussian policies are a typical choice for policy search methods [Deisenroth et al., 2013], and have been used in diverse approaches such as parameter-exploring policy gradients [Rückstieß et al., 2010], CMA-ES [Hansen et al., 2003], PoWER [Kober and Peters, 2009], PI2 [Theodorou et al., 2010], and REPS [Peters et al., 2010].

comparing the logarithms of the left- and right hand side of Eq. (4.4). For the left hand side, we obtain the simple expression

$$\log\left(\frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta}')}\right) = -\frac{\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta}}{2} + \boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\mu} + \frac{\boldsymbol{\theta}'^T\boldsymbol{\Lambda}\boldsymbol{\theta}'}{2} - \boldsymbol{\theta}'^T\boldsymbol{\Lambda}\boldsymbol{\mu}. \tag{4.6}$$

For the right hand side of (4.4), we can insert $g(\boldsymbol{\theta}'|\boldsymbol{\theta}) = N((1-\beta)\boldsymbol{\theta} + \beta\boldsymbol{\mu}, \tilde{\boldsymbol{\Lambda}}^{-1})$, with $\tilde{\boldsymbol{\Lambda}} = f(\beta)^{-2}\beta^{-2}\boldsymbol{\Lambda}$, and vice versa for $g(\boldsymbol{\theta}|\boldsymbol{\theta}')$. The resulting log-ratio is given as

$$\begin{aligned}
\log\left(\frac{g(\boldsymbol{\theta}|\boldsymbol{\theta}')}{g(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right) =& -\frac{\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}}{2} - (1-\beta)\beta\boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} - \frac{(1-\beta)^2\boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}'}{2} + \beta\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} \\
&+ \frac{\boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}'}{2} + (1-\beta)\beta\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} + \frac{(1-\beta)^2\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}}{2} - \beta\boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\mu \\
=& \left(2\beta-\beta^2\right)\left(-\tfrac{1}{2}\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta} + \boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} + \tfrac{1}{2}\boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}' - \boldsymbol{\theta}'^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu}\right) \\
=& \frac{2\beta-\beta^2}{f(\beta)^2\beta^2}\log\left(\frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta}')}\right),
\end{aligned}$$

where we inserted Eq. (4.6) in the last line. Now, we can identify, that for

$$f(\beta)^2 = (2\beta-\beta^2)/\beta^2 = 2/\beta - 1,$$

detailed balance is satisfied. Thus, $\tilde{\boldsymbol{\Lambda}}^{-1} = (2\beta-\beta^2)\boldsymbol{\Lambda}^{-1}$.

In principle, such generalized exploration can be used with different kinds of policy search methods. However, integrating coherent exploration might require minor changes in the algorithm implementation. In the following two sections, we will consider two types of methods: policy gradient methods and relative entropy policy search.

4.2.1 Generalized Exploration for Policy Gradients

In policy gradient methods, as the name implies, the policy parameters are updated by a step in the direction of the estimated gradient of the expected return over T time steps $J_\mu = \mathbb{E}\left[\sum_{t=0}^{T-1} r(\mathbf{s}_t, \mathbf{a}_t)\right]$ with respect to the meta-parameters $\boldsymbol{\mu}$ that govern the *distribution* over the policy parameters $\boldsymbol{\theta} \sim p_0 = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$.

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \alpha\nabla_\mu J_\mu,$$

with α a user-specified learning rate [Williams, 1992]. The gradient $\nabla_\mu J_\mu$ can be determined from the gradient of the log-policy [Williams, 1992, Baxter and Bartlett, 2001]

$$\nabla_\mu J_\mu = \mathbb{E}\left[\sum_{j=0}^{T-1}\Psi_j\left(\sum_{t=j}^{T-1} r_t - b_j\right)\right], \quad \Psi_j = \begin{cases} \nabla_\mu \log\pi_\mu(\mathbf{a}_0|\mathbf{x}_0) & \text{if } j=0, \\ \nabla_\mu \log\pi_\mu(\mathbf{a}_t|\mathbf{x}_t, \mathbf{a}_{t-1}) & \text{if } j>0, \end{cases}$$

considering that the action can depend on the previous action when using the generalized exploration algorithm. In these equations, b is a baseline that can be chosen to reduce the variance. Here, we will use the form of policy proposed in Equations (4.1)-(4.2), such that the controller selects parameters $\boldsymbol{\theta}$ independently of the state. In this case, Ψ_j is given by

$$\Psi_j = \begin{cases} \nabla_{\boldsymbol{\mu}} \log p_{0,\boldsymbol{\mu}}(\boldsymbol{\theta}_0) & \text{if } j = 0, \\ \nabla_{\boldsymbol{\mu}} \log p_{\boldsymbol{\mu}}(\boldsymbol{\theta}_t | \boldsymbol{\theta}_{t-1}) & \text{if } j > 0. \end{cases}$$

When $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ is chosen such that $\boldsymbol{\theta}_t$ is independent of $\boldsymbol{\theta}_{t-1}$, the resulting algorithm is known as G(PO)MDP [Baxter and Bartlett, 2001]. Choosing $p_{\boldsymbol{\mu}}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = \delta(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t)$ on the other hand results in the PEPG algorithm proposed by Sehnke et al. [Sehnke et al., 2010].

In our paper, we will focus on learning the mean $\boldsymbol{\mu}$ of a Gaussian policy $\mathcal{N}(\mathbf{a}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In that case, the relevant gradients are given by

$$\nabla_{\boldsymbol{\mu}} \log(p_0(\boldsymbol{\theta}_0; \boldsymbol{\mu}, \boldsymbol{\Sigma})) = \nabla_{\boldsymbol{\mu}} \log(\mathcal{N}(\boldsymbol{\theta}_0; \boldsymbol{\mu}, \boldsymbol{\Sigma})) = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}_0 - \boldsymbol{\mu}),$$

for the initial policy, and

$$\begin{aligned} \nabla_{\boldsymbol{\mu}} \log(p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}; \boldsymbol{\mu}, \boldsymbol{\Sigma})) &= \nabla_{\boldsymbol{\mu}} \log \mathcal{N}\left(\boldsymbol{\theta}_t; \beta\boldsymbol{\mu} + (1-\beta)\boldsymbol{\theta}_{t-1}, (2\beta - \beta^2)\boldsymbol{\Sigma}\right) \\ &= (2-\beta)^{-1}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}_t - \beta\boldsymbol{\mu} - (1-\beta)\boldsymbol{\theta}_{t-1}), \end{aligned}$$

for the proposal distribution. We can now easily verify that for $\beta = 1$, $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p_0(\boldsymbol{\theta}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, making the algorithm identical to G(PO)MDP, whereas for $\beta = 0$, $\nabla_{\boldsymbol{\mu}} \log p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = 0$, since $\boldsymbol{\theta}_{t-1} = \boldsymbol{\theta}_t$ in this case. This property makes the algorithm equivalent to PEPG for $\beta = 0$. Setting $0 < \beta < 1$ yields intermediate strategies that trade off the advantages of G(PO)MDP and PEPG.

4.2.2 Generalized Exploration for Relative Entropy Policy Search

In relative entropy policy search (REPS), the goal is to take larger steps than policy gradient methods while staying close to the previous sampling policy in information-theoretic terms [Peters et al., 2010, van Hoof et al., 2015b]. This objective is reached by solving the optimization problem

$$\max_{\pi, \mu_\pi} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})r(\mathbf{s},\mathbf{a})d\mathbf{a}d\mathbf{s}, \tag{4.7}$$

$$\text{s. t.} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})d\mathbf{a}d\mathbf{s} = 1, \tag{4.8}$$

$$\forall s'. \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})d\mathbf{a}d\mathbf{s} = \mu_\pi(\mathbf{s}'), \tag{4.9}$$

$$\mathrm{KL}(\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})||q(\mathbf{s},\mathbf{a})) \leq \beta, \tag{4.10}$$

where $\mu_\pi(\mathbf{s})$ is the steady-state distribution under $\pi(\mathbf{a}|\mathbf{s})$, as enforce by Eq. (4.9), and $\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ is the reward-maximizing distribution as specified by Eqs. (4.7-4.8).

Equation (4.10) specifies the additional information-theoretic constraints, where q is a reference distribution (e.g. the previous sampling distribution), and KL denotes the KL divergence [Peters et al., 2010]. The expected values in Eqs. (4.7-4.10) are approximated using the sampled data.

The solution to Eqs. (4.7-4.10) is a re-weighting $w(\mathbf{s}, \mathbf{a})$ of the reference distribution q, with $\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s}) = w(\mathbf{s}, \mathbf{a})q(\mathbf{s}, \mathbf{a})$, as derived in detail in [Peters et al., 2010]. To find this weighting in continuous-state systems, we need to approximate the steady-state constraint[2] in Eq. (4.8) as

$$\iint_{\mathcal{S}\times\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})p(\mathbf{s}'|\mathbf{s}, \mathbf{a})\boldsymbol{\phi}(\mathbf{s}')d\mathbf{a}d\mathbf{s}d\mathbf{s}' = \int_{\mathcal{S}} \mu_\pi(\mathbf{s}')\boldsymbol{\phi}(\mathbf{s}')d\mathbf{s}'. \tag{4.11}$$

using features ϕ of the state. Since we will look at deterministic dynamical systems[3], the expected features under the transition distribution $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ are simply given by the subsequent state in the roll-out [Peters et al., 2010].

The re-weighting coefficients $w(\mathbf{s}, \mathbf{a})$ can only be calculated at sampled state-action pairs $(\mathbf{s}, \mathbf{a})$. To find a generalizing policy that is defined at all states, the sample-based policy can be generalized by optimizing a maximum likelihood objective

$$\arg\max_{\boldsymbol{\mu},\mathbf{D}} \prod_i^M L_i(\boldsymbol{\mu}, \mathbf{D}), \quad L_i(\boldsymbol{\mu}, \mathbf{D}) = p\left(\mathbf{a}_{1:N}^{(i)} \middle| \mathbf{s}_{1:N}^{(i)}; \boldsymbol{\mu}, \mathbf{D}\right), \tag{4.12}$$

where $\mathbf{s}_{1:N}^{(i)}$ is the sequence of states encountered in episode $i \in \{1, \ldots, M\}$, and $\mathbf{a}_{1:N}^{(i)}$ are the corresponding actions. The hyper-parameters, consisting of $\boldsymbol{\mu}$ and the entries of diagonal covariance matrix $\mathbf{D}$, govern a distribution $p(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathbf{D})$ over policy parameters $\boldsymbol{\theta}$ for policies of the form $\mathbf{a} = \boldsymbol{\phi}(s)^T\boldsymbol{\theta}$. Earlier work has focused on the case where actions during an episode are chosen independently of each other [Peters et al., 2010, van Hoof et al., 2015b]. However, with coherent exploration, policy parameters are similar in subsequent time-steps and, thus, this assumption is violated. Here, instead we define the likelihood terms as

$$L_i = \int_{\Theta^N} p(\mathbf{a}_{1:N}^{(i)}|\mathbf{s}_{1:N}^{(i)}, \boldsymbol{\theta}_{1:N}^{(i)})p(\boldsymbol{\theta}_{1:N}^{(i)}|\boldsymbol{\mu}, \mathbf{D})d\boldsymbol{\theta}_{1:N}^{(i)}, \tag{4.13}$$

with

$$p\left(\mathbf{a}_{1:N}^{(i)} \middle| \mathbf{s}_{1:N}^{(i)}, \boldsymbol{\theta}_{1:N}^{(i)}\right) = \mathcal{N}\left(\boldsymbol{\phi}\left(\mathbf{s}_{1:N}^{(i)}\right)^T\boldsymbol{\theta}_{1:N}^{(i)}, \sigma^2\mathbf{I}\right), \tag{4.14}$$

where $\boldsymbol{\theta}_{1:N}^{(i)}$ denote the sequence of parameters explored over the N time steps of the ith episode. Under the proposal distribution of Eq. (4.5), the distribution over these parameters is given by

$$p\left(\boldsymbol{\theta}_{1:N}^{(i)}|\boldsymbol{\mu}, \mathbf{D}\right) = p\left(\boldsymbol{\theta}_1^{(i)} \middle| \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\prod_{j=1}^N p\left(\boldsymbol{\theta}_j^{(i)} \middle| \boldsymbol{\theta}_{j-1}^{(i)}, \boldsymbol{\mu}, \mathbf{D}\right) = \mathcal{N}\left(\boldsymbol{\theta}_{1:N}^{(i)} \middle| \tilde{\boldsymbol{\mu}}, \mathbf{D} \otimes \mathbf{E}\right) \tag{4.15}$$

[2] This approximation is equivalent to approximating the function-valued Lagrangian multiplier for the continuum of constraints in Eq. (4.9) by a function linear in the features ϕ [van Hoof et al., 2015b].

[3] For stochastic systems, we could approximate the expected features using a learned transition model as proposed by van Hoof et al. [2015b].

In this equation, $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}^T, \ldots, \boldsymbol{\mu}^T]^T$, $[\mathbf{E}]_{jk} = (1-\beta)^{|j-k|}$, and $\otimes$ denotes the Kronecker product. Inserting (4.14) and (4.15) into (4.13) yields the equation

$$L_i = \mathcal{N}\left(\mathbf{a}_{1:N}^{(i)} \middle| \boldsymbol{\phi}(\mathbf{s}_{1:N}^{(i)})^T \boldsymbol{\mu}, \boldsymbol{\Sigma}^{(i)} + \sigma^2 I\right), \tag{4.16}$$

where the elements of the covariance matrix are given by

$$\boldsymbol{\Sigma}_{jk}^{(i)} = \sigma^2 \boldsymbol{\phi}(\mathbf{s}_j)^T \boldsymbol{\phi}(\mathbf{s}_k)(1-\beta)^{|j-k|}.$$

However, this section has so far assumed we have samples $(\mathbf{a}_j, \mathbf{s}_j) \sim \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$, whereas we only have access to samples $(\mathbf{a}_j, \mathbf{s}_j) \sim q$ from the sampling distribution and re-weighting factors $w_j = p(a_j, s_j)/q(a_j, s_j)$. We can use importance weighting, meaning that we maximize the weighted log-likelihood

$$\sum_{i=1}^{M}\sum_{j=1}^{N} w_j^{(i)} \log \mathcal{N}\left(\mathbf{a}_j^{(i)}; \boldsymbol{\phi}\left(\mathbf{s}_{1:N}^{(i)}\right)^T \boldsymbol{\mu}, \boldsymbol{\Sigma}_{jj} + s^2\right)$$

Weighting the samples is equivalent to scaling the variance matrix up to a proportionality constant. Since $\boldsymbol{\Sigma}_{jk} = \rho_{jk}\sqrt{\boldsymbol{\Sigma}_{jj}\boldsymbol{\Sigma}_{kk}}$, with ρ_{jk} the correlation coefficient, re-scaling $\boldsymbol{\Sigma}_{jj}$ by $1/w_j$ means that $\boldsymbol{\Sigma}_{jk}$ has to be scaled by $1/\sqrt{w_j}$ accordingly, such that we define

$$\tilde{\boldsymbol{\Sigma}}_{jk}^{(i)} = \sigma^2 \boldsymbol{\phi}(\mathbf{s}_j)^T \boldsymbol{\phi}(\mathbf{s}_k)(1-\beta)^{|j-k|}\left(w_j^{(i)} w_k^{(i)}\right)^{-\frac{1}{2}}.$$

We can now solve $\arg\max_{\boldsymbol{\mu}} \prod_i L_i$ in closed form, yielding

$$\boldsymbol{\mu}^* = \left(\sum_{i=1}^{M} \boldsymbol{\phi}\left(\mathbf{s}_{1:N}^{(i)}\right) \tilde{\boldsymbol{\Sigma}}^{(i)} \boldsymbol{\phi}\left(\mathbf{s}_{1:N}^{(i)}\right)^T\right)^{-1} \sum_{i=1}^{M} \boldsymbol{\phi}\left(\mathbf{s}_{1:N}^{(i)}\right) \tilde{\boldsymbol{\Sigma}}^{(i)} \mathbf{a}_{1:N}^{(i)}. \tag{4.17}$$

However, there is no closed-form solution for the elements of $\mathbf{D}$, so we solve

$$\mathbf{D}^* = \arg\max_{\mathbf{D}} \prod_{i=1}^{M} p\left(\mathbf{a}_{1:N}^{(i)} \middle| \mathbf{s}_{1:N}^{(i)}; \boldsymbol{\mu}^*, \mathbf{D}\right),$$

by using a numerical optimizer. The variance σ^2 of the action likelihood term in Eq. (4.14) is set to 1 in our experiments. This variance is small relative to the maximum action, and acts as a regularizer in Eq. (4.17).

4.3 Experiments and Results

In this section, we employ the generalized exploration algorithms outlined above to solve different reinforcement learning problems with continuous states and actions. In our experiments, we want to show that generalized exploration can be used to obtain better policies than either the step-based or the episode-based exploration approaches found in prior work. We will also look more specifically into some of the factors mentioned in Section 4.1 that can explain some of the differences in performance. First, we evaluate generalized exploration in a policy gradient algorithm on a linear control task. Then, we will evaluate generalized exploration in relative entropy policy search on two tasks: an inverted pendulum balancing task with control delays and an underpowered pendulum swing-up task.

4.3.1 Policy Gradients in a Linear Control Task

In the first experiment, we consider a dynamical system where the state $\mathbf{s} = [x, \dot{x}]^T$ is determined by the position and velocity of a point mass of $m = 1\text{kg}$. The initial state of the mass is distributed as a Gaussian distribution with $x \sim \mathcal{N}(-7.5, 5^2)$ and $\dot{x} \sim \mathcal{N}(0, 0.5^2)$. The position and velocity of the mass are limited to $-20 \leq x \leq 20$, $-10 \leq \dot{x} \leq 10$. The goal of the controller to bring the mass to the phase-space origin is defined by the reward function

$$r(\mathbf{s}) = -\frac{3}{200}\mathbf{s}^T\mathbf{s} + \exp\left(-\frac{\mathbf{s}^T\mathbf{s}}{8}\right).$$

As action, a force can be applied to this mass. Furthermore, friction applies a force of $-0.5\dot{x}\text{N}$. The actions are chosen according to the linear policy $a = \boldsymbol{\theta}^T\mathbf{s}$, with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, 1)$, where $\boldsymbol{\mu}$ is initialized as $\mathbf{0}$ and subsequently optimized by the policy gradient algorithm outlined in Section 4.2.1. Every episode consists of 50 time-steps of 0.1 s. As baseline for the policy gradients, a function quadratic in the state is chosen. Its parameters are optimized to minimize the gradient's variance, as detailed in 4.A.

The rationale for this task is, that it is one of the simplest task where consistent exploration is important, since the second term in the reward function will only yield non-negligible values as the point mass gets close to the origin. Our proposed algorithm is a generalization of the the G(PO)MDP and PEPG algorithms, we obtain those algorithms if we choose $\beta = 1$ (GPOMDP) or $\beta = 0$ (PEPG). We will compare those previous algorithms to other settings for the exploration coherence term β. Besides analyzing the performance of the algorithms in terms of average reward, we will look at how big a range of positions is explored by the initial policy for the various settings.

For every condition, 20 trials were performed. In each trial, 40 iterations were performed that consist of a data-gathering step and a policy update step. Seven episodes were performed in each iteration, as seven is the minimum number of roll-outs required to fit the baseline parameters in a numerically stable manner. As different values of the coherence parameter β require a different step size for optimal performance within the 40 available iterations, we ran each condition for each step size $\alpha \in \{0.2, 0.15, 0.10, 0.05, 0.025\}$, and use the step-size that yielded maximal final average reward.

4.3.2 Results and Discussion of the Linear Control Experiment

The results of the linear control experiment are shown in Figures 4.1 and 4.2. The average rewards obtained by the systems are shown in Figure 4.1, where the best step size α for each value of the trade-off parameter β is shown. In this figure, we can see that for intermediate values of the temporal coherence parameter β, the learning speed tends to be faster in the first 100 roll-outs. Furthermore, for $\beta = 0.15$ or $\beta = 0.05$, best performance is obtained at the end of the experiment. Suboptimal performance for PEPG ($\beta = 0$) can be caused by the fact that PEPG can only try a number of parameters equal to the number of roll-outs per iteration, which can lead

to high-variance updates. Suboptimal performance for G(PO)MDP ($\beta = 1$) can be caused by the 'washing out' due to the high frequency of policy perturbations.

To investigate this possible cause, in Figure 4.2, we show example trajectories as well as the evolution of the standard deviation of the position x. In Figure 4.2a, example trajectories under the initial policies are shown. Here, the difference between coherent exploration and high-frequency perturbations are clearly visible. Figure 4.2b shows that, from the initial standard deviation, low values of β yield a higher increase in variance over time, indicating those variants explore more of the state-space. This difference is likely to be caused by those methods exploring different 'strategies' that visit different parts of the state-space, rather than the high-frequency perturbations for high β that tend to induce random walk behavior. The distribution of positions cannot grow indefinitely, as the position limits of the systems are reached. This limit explains why the distributions does not keep growing as fast in later time-steps.

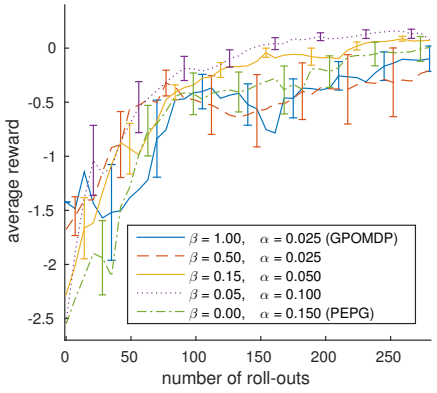
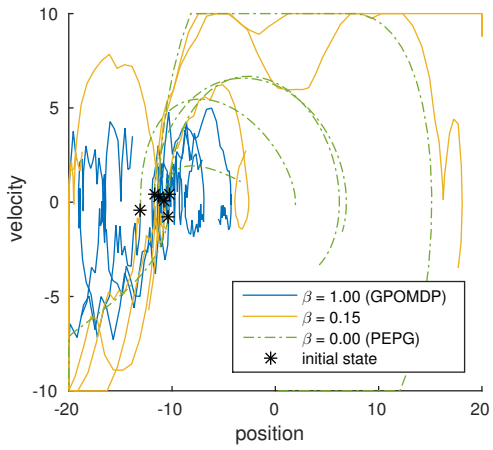


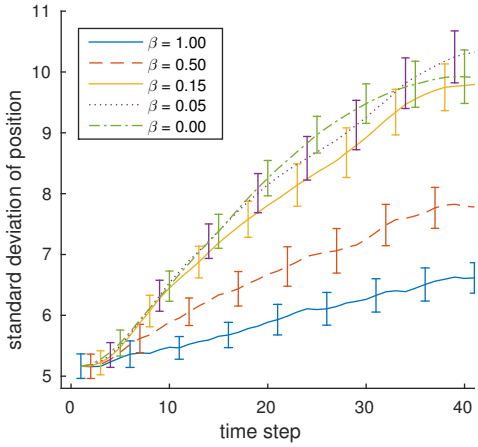
Figure 4.1.: Average reward in the linear control task with policy gradient methods. Error bars show the standard error over 20 trials.

4.3.3 REPS for Inverted Pendulum Balancing with Control Delays

In this experiment, we consider the task of balancing a pendulum around its unstable equilibrium by applying torques at its fulcrum. The pendulum we consider has a mass $m = 10$kg and a length $l = 0.5$m. Furthermore, friction applies a force of



(a) Example trajectories under different settings of the coherency parameter.



(b) Standard deviation of positions reached. Error bars show the standard error over 20 trials.

Figure 4.2.: Example trajectories and distribution statistics under the initial policy using G(PO)MDP ($\beta = 1$) and PEPG ($\beta = 0$) as well as other settings for β.

$0.36\dot{x}$Nm. The pendulum's state is defined by its position and velocity $\mathbf{s} = [x, \dot{x}]^T$, where the angle of the pendulum is limited $-1 < x < 1$. The chosen action $-40 < a < 40$ is a torque to be applied at the fulcrum for a time-step of 0.05s second. However, in one of our experimental conditions, we simulate control delays of 0.025s, such that the actually applied action is $0.5a_t + 0.5a_{t-1}$. This condition breaks the Markov assumption, and we expect that smaller values of the trade-off parameter β will be more robust to this violation. The action is chosen according to a linear policy $a = \boldsymbol{\theta}^T \mathbf{s}$. The parameters are chosen from a normal distribution $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{D})$, which is initialized using $\boldsymbol{\mu} = \mathbf{0}$, and D a diagonal matrix with $D_{11} = 120^2$ and $D_{22} = 9^2$. Subsequently, $\boldsymbol{\mu}$ and $\mathbf{D}$ are updated according to the generalized REPS algorithm introduced in Section 4.2.2. We use the quadratic reward function $r(x, \dot{x}, a) = 10x^2 + 0.1\dot{x}^2 + 0.001a^2$.

Roll-outs start at a position $x \sim \mathcal{N}(0, 0.2^2)$ with a velocity $\dot{x} \sim \mathcal{N}(0, 0.5^2)$. At every step, there is a fixed probability of 10% of terminating the episode [van Hoof et al., 2015b]. As such, each episode contains 10 time steps on average. Initially, 60 roll-outs are performed. At every iteration, the 20 oldest roll-outs are replaced by new samples. Then, the policy is updated using these samples. The sampling distribution q is, thus, a mixture of state-action distributions under the previous three policies. For the features ϕ_i in Eq. (4.11), we use 100 random features that approximate the non-parametric basis in [van Hoof et al., 2015b]. These random features $\boldsymbol{\Phi}$ are generated according to the procedure in [Rahimi and Recht, 2007], using manually specified bandwidth parameters, resulting in

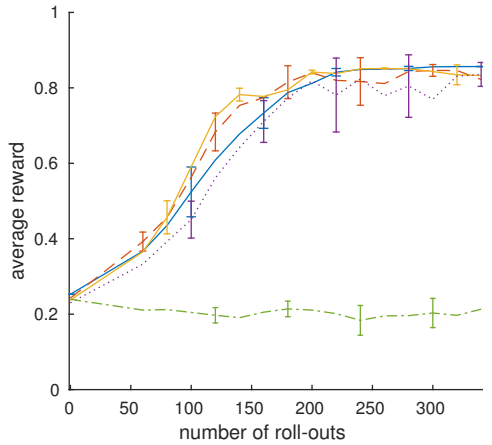
$$\phi_i(\mathbf{s}) = 50^{-1/2} \cos\left([\cos(x), \sin(x), \dot{x}]\boldsymbol{\omega}_i + b_i\right), \tag{4.18}$$

where b is a uniform random number $b \in [0, 2\pi]$ and $\omega_i \sim \mathcal{N}(\mathbf{0}, \mathbf{B}^{-1})$, where $\mathbf{B}$ is a diagonal matrix with the squared kernel bandwidth for each dimension. In our experiments, the bandwidths are 0.35, 0.35, and 6.5, respectively.

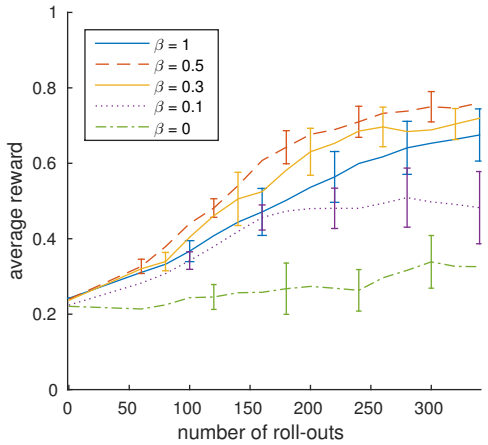
In our experiment, we will compare different settings of the coherence parameter β under a condition without delays and a condition with the half time-step delay as explained earlier in this section. In this condition, we want to test the assumption that a lower value of β makes the algorithm more robust against non-Markov effects. For $\beta = 1$, we obtain the algorithm described in [Peters et al., 2010, van Hoof et al., 2015b]. We will compare this previous step-based REPS algorithm to other settings of the coherence trade-off term β.

4.3.4 Results of the Pendulum Balancing Experiment

The results of the inverted pendulum balancing task are shown in Figure 4.3. The results on the standard balancing task, without control delays, are shown in 4.3a. This figure shows that, generally, values of the consistency trade-off parameter β of at least 0.3 result in better performance than setting $\beta = 0.1$. Setting $\beta = 0$ results in the algorithm being unable to improve the policy. Being able to try only one set of parameters per roll-out could be one cause, but the procedure described in Section 4.2.2 might also struggle to find a distribution that matches all weighted samples while keeping the parameter values constant for the entire trajectory. Between the



(a) Inverted pendulum balancing without delays.



(b) Inverted pendulum balancing with a delay of half a time-step.

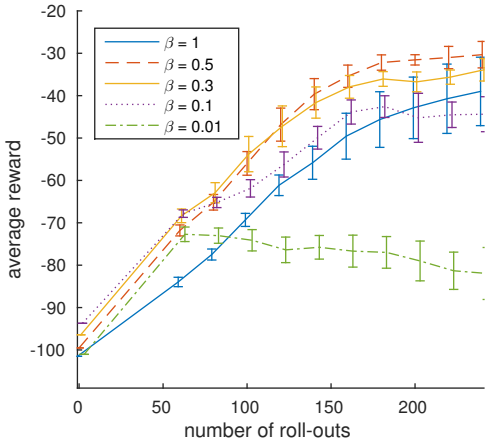
Figure 4.3.: Inverted pendulum balancing tasks with control delays using relative entropy policy search. Error bars show twice the standard error over 10 trials, and are shown at selected iterations to avoid clutter.

different settings with $\beta \geq 0.3$ small differences exist, possibly because the standard version of REPS with time-independent exploration ($\beta = 1$) suffers from 'washing out' of exploration signals like in the policy gradient experiment in Section 4.3.1.

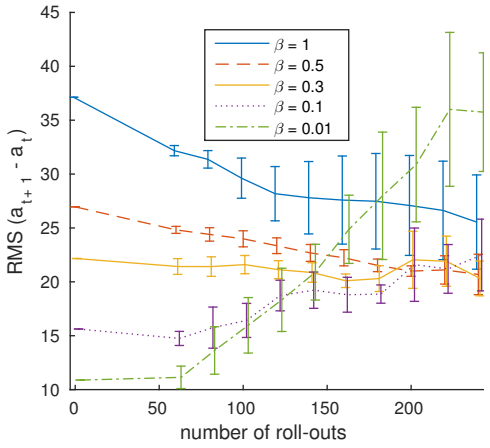
In a second experimental condition, we simulate control delays, resulting in the applied action in a certain time step being a combination of the actions selected in the previous and current time steps. This violation of the Markov assumption makes the task harder. As expected, Figure 4.3b shows that the average reward drops for all conditions. For $\beta = 1$, the decrease in performance is much bigger than for $\beta = 0.5$ or $\beta = 0.3$. However, unexpectedly $\beta = 0.5$ seems to yield better performance than smaller values for the trade-off parameter. We suspect this effect to be caused by a too sparse exploration of the state-action space for each set of policy parameters, together with a possible difficulty in maximizing the resulting weighted likelihood as discussed in the previous paragraph.

4.3.5 REPS for Underpowered Swing-up

In the underpowered swing-up task, we use the same dynamical system as in the previous experiment with the following modifications: the pendulum starts hanging down close to the stable equilibrium at $x = \pi$, with $x_0 \sim \mathcal{N}(\pi, 0.2^2)$ with $\dot{x}_0 = 0$. The episode was re-set with a probability of 2% in this case, so that the average episode length is fifty time steps. The pendulum position is in this case not limited, but projected on $[-0.5\pi, 1.5\pi]$. Actions are limited between -30 and 30N. A direct swing-up is consequently not possible, and the agent has to learn to make a counter-swing to gather momentum first. Since a linear policy is insufficient, instead, we use a policy linear in exponential radial basis features with a bandwidth of 0.35 in the position domain and 6.5 in the velocity domain, centered on a 9×7 grid in the state space, yielding 63 policy features. Optimizing 63 entries of the policy variance matrix



(a) Average reward for the underpowered swing-up task.



(b) Root-mean square of the difference between subsequent actions indicates applied jerks.

Figure 4.4.: Pendulum swing-up task using relative entropy policy search. Error bars show twice the standard error over 10 trials, and are slightly offset to avoid clutter.

$\mathbf{D}$ would slow the learning process down drastically, so in this case we used a spherical Gaussian with $\mathbf{D} = \lambda\mathbf{I}$, so that only a single parameter λ needs to be optimized.

Besides evaluating the average rewards obtained using different values of the exploration coherence trade-off term β, we evaluate the typical difference between subsequent actions as measured by the root-mean-squared difference between subsequent actions. Actions correspond to applied torques in this system, and the total torque (from applied actions and gravity) is directly proportional to the rotational acceleration. Thus, a big difference in subsequent actions can cause high jerks, which causes wear and tear on robotic systems. As such, in most real systems we would prefer the typical difference between subsequent actions to be low.

4.3.6 Results of the Underpowered Swing-up Experiment

The results on the pendulum swing-up task are shown in Figure 4.4. With $\beta = 0$, policies quickly deteriorated, since the 60 episodes used in each iteration did not allow the 63-dimensional policies to be fitted. Therefore, this condition is not shown in the figures. For $\beta = 0.01$, the poor performance is likely due to the same cause, although the difficulty of matching the weighted samples discussed in the pendulum balancing experiment might play a role as well. For other values of β, Figure 4.4a shows, that setting the trade-off parameter to an intermediate value yields higher learning speeds than setting $\beta = 1$ as in the original REPS algorithm. Again, the washing out of high-frequency exploration could be a cause of this effect.

Figure 4.4b shows another benefit of setting the exploration coherence parameter to an intermediate value. The typical difference between chosen actions is more than 50% higher initially for the original REPS algorithm ($\beta = 1$), compared to setting $\beta = 0.3$. This behavior will cause higher jerks, and thus more wear and tear, on robot systems where these controllers are applied. The typically higher difference

between actions persist even as the algorithm gets close to an optimal solution after 15 iterations. The case of $\beta = 0.01$ goes against this trend, as this variant tends to induce very high differences between subsequent actions because of the extremely high values for λ returned by the optimizer. These high values could be caused by the difficulty to match weighted samples as discussed before.

4.4 Conclusion

To conclude this chapter, first, we first give a summarize the presented contributions and experiments. Then, we discuss potential future work in the epilogue.

4.4.1 Summary of this Chapter

In this chapter, we introduce a generalization of step-based and episode-based exploration of controller parameters. This generalization allows different trade-offs to be made between the advantages and disadvantages of temporal coherence in exploration. Whereas independent perturbation of actions at every time step allows more parameter values to be tested within a single episode, fully coherent (episode-based) exploration has the advantages of, among others, avoiding 'washing out' of explorative perturbations, being robust to non-Markovian aspects of the environment, and inducing lower jerk, and thus less strain, on experimental platforms.

Our experiments confirm these advantages of coherent exploration, and show that intermediate strategies between step-based and episode-based exploration provide some of the same advantages. In terms of average reward, as expected, for many systems intermediate trade-offs between completely independent, step-based exploration and complete correlated, episode exploration, provides the best learning performance.

4.4.2 Epilogue

In this chapter, a novel exploration coherence trade-off has been introduced, yielding a generalizing algorithm that includes earlier time-step-based and episode-based exploration algorithms as special cases. Sections 4.1–4.4.1 of this chapter have been submitted to the Machine Learning journal [van Hoof and Peters, 2016]. A couple of promising extensions and applications of this work are to be addressed in future work.

Many of the benefits of coherent exploration, such as robustness to non-Markov effects and lower jerk during roll-outs, are especially important on robotic systems. Therefore, we want to investigate the performance of generalized exploration algorithms in real dynamical systems.

So far, we have only investigated using generalized exploration in fully observable Markov Decision processes. However, many real control tasks are partially observable to some degree. Applying similar techniques for generalized exploration in partially observable Markov decision processes (POMDPs) could bring similar benefits to this wider class of problems.

One shortcoming of the proposed method is, that smoothly changing the parameters only results in smooth changes in behavior, if the underlying features change smoothly. In environments where the support of features needs to be small relative to the domain, even the proposed approach cannot guarantee coherent exploration.

Methods that sample a transition model [Strens, 2000, Asmuth et al., 2009] or value function [Osband et al., 2016] for every episode do obtain coherent behavior even for non-smooth features, by directing exploration to a global goal. Such methods, like coherent exploration obtained through keeping the policy parameters constant, can only evaluate one hypothesis per roll-out. A promising idea for future work is to slowly change the sampled objects (transition models or value function) in the manner proposed for the policy in this chapter. This combined approach could yield better exploration coherence trade-offs even with non-smooth features.

The PEGASUS method [Ng and Jordan, 2000] can be used to transform any stochastic MDP into a deterministic MDP. This approach can be used to find lower-variance estimates of policy gradients, especially if episode-based evaluation is used. Therefore, in future work we want to investigate the possibility of integrating PEGASUS-style policy search with coherent exploration. We expect this combination to yield potentially large benefit in the highly coherent exploration regime.

4.A Derivation of Baseline with State Features

In our approach, we use a state-dependent variance-minimizing baseline $b_k(\mathbf{s}_t) = \boldsymbol{\alpha}_k^T \phi(\mathbf{s}_t)$, with ϕ quadratic state features. We derive this baseline following the procedure in [Deisenroth et al., 2013, Peters and Schaal, 2008a]. First, the variance of the k-th dimension of the policy gradient is expressed as

$$\mathbb{E}\left[\left(\nabla_{\mu_k}\sum_{j=1}^{T}\Psi_j\left(\sum_{t=j}^{T}r_t-\boldsymbol{\alpha}_k^T\phi(\mathbf{s}_j)\right)\right)^2\right]-\mathbb{E}\left[\nabla_{\mu_k}\sum_{j=1}^{T}\Psi_j\left(\sum_{t=j}^{T}r_t-\boldsymbol{\alpha}_k^T\phi(\mathbf{s}_j)\right)\right]^2,$$

where

$$\Psi_j=\begin{cases}\nabla_\mu \log p_{0,\mu}(\boldsymbol{\theta}_0) & \text{if } j=0,\\ \nabla_\mu \log p_\mu(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) & \text{if } j>0.\end{cases}$$

The second term is not affected by the baseline, since

$$\mathbb{E}\left[\nabla_{\mu_k}\sum_{j=1}^{T}\Psi_j\boldsymbol{\alpha}_k^T\phi(s_j)\right]=\sum_{j=1}^{T}\boldsymbol{\alpha}_k^T\phi(\mathbf{s}_j)\nabla_{\mu_k}\int_\Theta p_\mu(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})d\boldsymbol{\theta}=0.$$

Taking the derivative of the first term with respect to $\boldsymbol{\alpha}_k$ and setting it to 0 to find the minimum yields

$$\begin{aligned}\nabla_\alpha\mathbb{E}\left(\sum_{j=1}^{T}\Psi_j\left(\sum_{t=j}^{T}r_t-\boldsymbol{\alpha}_k^T\phi(\mathbf{s}_j)\right)\right)^2 &\stackrel{!}{=} \mathbf{0},\\ -2\mathbb{E}\left(\sum_{j=1}^{T}\Psi_j\left(\left(\sum_{t=j}^{T}r_t\right)-\boldsymbol{\alpha}_k^T\phi(\mathbf{s}_j)\right)\right)\left(\sum_{j=1}^{T}\Psi_j\phi(\mathbf{s}_j)\right) &= \mathbf{0}.\end{aligned}$$

because of the cross-dependencies between the entries of $\boldsymbol{\alpha}$, this system is easiest solved as a matrix equation, with $\mathbf{J}_j=\sum_t^j r_t$,

$$\begin{aligned}\mathbf{0} &= -2\mathbb{E}\left(\boldsymbol{\Psi}^T\mathbf{J}-\boldsymbol{\alpha}^T\boldsymbol{\Phi}^T\boldsymbol{\Psi}\right)\boldsymbol{\Psi}^T\boldsymbol{\Phi},\\ \mathbb{E}\left[\boldsymbol{\Psi}^T\mathbf{J}\boldsymbol{\Psi}^T\Phi\right] &= \boldsymbol{\alpha}\mathbb{E}\left[\boldsymbol{\Phi}^T\boldsymbol{\Psi}\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right],\\ \boldsymbol{\alpha} &= \mathbb{E}\left[\boldsymbol{\Phi}^T\boldsymbol{\Psi}\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right]^{-1}\mathbb{E}\left[\boldsymbol{\Psi}^T\mathbf{J}\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right].\end{aligned}$$

5 Conclusion and Future Work

The research described in this thesis aims at developing machine learning methods that allow robots to learn about novel environments or obtain new skills, while reducing the amount of human input needed in the learning process. Thus, the methods described in this thesis rely on the robot's capability to explore its environment rather than human demonstrations or annotations. Furthermore, where possible I tried to minimize or avoid the need for manual tweaking and feature design.

Specifically, in Chapter 2, I investigated the problem of interactive perception, in other words, how the robot can use its actions to discover hidden or perceptually ambiguous properties of the environment. By using its own actions to obtain information, the robot does not rely on provided annotations or demonstrations. The non-parametric Bayesian approach allowed the robot to integrate noisy clues from different modalities, and avoids having to manually tune parameter values or specify the number of objects in the scene. Furthermore, samples from the posterior distribution represent the robot's uncertainty, which allows the robot to choose explorative actions that reduce this uncertainty. My experiments show, that choosing actions in this manner makes the learning process more data-efficient.

After that, in Chapter 3, I focus on making robots learn reward-maximizing behavior through reinforcement learning. Most conventional methods are designed for systems with discrete states, or for systems with carefully engineered feature representations. The approaches that avoid this issue using non-parametric methods have largely fallen in two categories. Some of them do not represent the control policy, which causes big jumps in the policy space as subsequent approximations of the value function are maximized. Other approaches rely on policy gradients; those methods tend to need impractically many samples as the learning rate is low. I combined the advantages of non-parametric reinforcement learning with an information-theoretical bound on the policy update, that is able to take larger steps than policy gradient methods while avoiding unstable behavior. By avoiding manually engineered features, the design effort is reduced. The addition of a learned transition model helps the algorithm cope with non-deterministic system behavior. Evaluations on, among others, a real-robot pendulum swing-up task with visual input show that the proposed algorithm can indeed learn challenging tasks with high-dimensional, redundant inputs.

In the last chapter of this thesis, a closer look is taken at the coherence of exploration in reinforcement learning methods that employ stochastic policies. Sampling actions or controller parameters at every time step independently, resulting in incoherent, high-frequency exploration. The disadvantage of that, is that a random walk behavior is obtained that explores the state-space inefficiently. Furthermore such schemes are sub-optimal for real-robot reinforcement learning, since they induce high jerks which might damage the robot and are fragile to delays in actuation and communication. Only applying perturbations at the beginning of the episode, however, might require more sample data. Instead, I propose a coherence trade-off parameter which can be

set to obtain step-based or episode-based exploration, or intermediate behavior that combines some of the advantages of both extremes. In the experiments, I show that intermediate behavior is beneficial across different tasks and different policy search methods. Furthermore, the results confirm some of the expected advantages of exploration coherence, such as more efficient exploration of the state space, robustness to delays, and lower jerks. Combining these advantages with a relatively high sample efficiency results in a trade-off that could make it easier to apply reinforcement learning methods to real-robot systems.

5.1 Future Work

The experiments in this thesis have been largely confined to situations where the underlying system had limited degrees of freedom. In such settings those methods are successful in allowing agents to learn about environments or obtain new skills, while reducing the design effort needed from a human designer. However, applying these algorithms to robots in human environments like homes or hospitals with a much larger number of degrees of freedom would still require an infeasible amount of data.

Therefore, one of the main challenges for future work is scaling methods for robot learning through exploration up to more complex tasks in larger environments. Several ways of reaching this goal are conceivable. First of all, making stronger use of the background knowledge about robots and their environments could speed up learning. In some cases, a lot of knowledge might be available, such as a dynamics and kinematics model as well as a map of the environment. In other cases, only general properties might be known, such that Newton's laws of motion apply, or that the robotic system is action-affine. In either of the two cases, parts of the transition dynamics are known whereas others are not (for example, the location of objects and their properties). Thus, methods should be investigated that allow trading of exploration and exploitation in an environment with both known and unknown aspects. Recent work has investigated such methods with promising results, but has largely focused on following pre-specified trajectories [Vuga et al., 2015, Englert and Toussaint, 2016, Rückert et al., 2013]. One possible alternative approach is belief space planning, where background knowledge can be included as prior information, and which allows optimally trading off exploration and exploitation.

As alternative or complementary approach, hierarchical reinforcement learning has been proposed to allow learning of more complex tasks in larger domains by breaking the tasks up into sub-policies or options that reach intermediate goals [Sutton et al., 1999b, Kaelbling, 1993, Parr and Russell, 1998, Dietterich, 2000]. However, many current methods either rely on pre-defined options, or use additional information such as the goal location, demonstration, or the transition dynamics to learn those options from data. Investigating whether and how options can be learned using more a more general criterion, such as maximizing the model evidence or minimizing the model description length, could make it possible to apply such these methods if such additional information is not present.

Lastly, the kind of exploration discussed in Chapter 4 allows to explore different strategies more globally, which is extremely helpful in large state spaces. However,

coherently changing policy parameters only makes the resulting behavior coherent if most parameters influence behavior globally. When a policy based on, for example, local radial basis functions is used, the coherence of the behavior would depend on the width of the basis function relative to the size of the state space. By combining the idea of slowly change rather than dithering with methods to sample value functions [Wyatt, 1998, Dearden et al., 1998, Osband et al., 2016] or transition models [Asmuth et al., 2009, Strens, 2000, Ortega and Braun, 2010, Dearden et al., 1999, Doshi-Velez et al., 2010], a novel method could be obtained that allows defining the exploration coherence independent of the type of policy parametrization that is used.

Another problem to be addressed to apply reinforcement learning methods in real-world environments is partial observability and observation noise. Most reinforcement learning methods assume the environment is a Markov process (i.e., the state of the environment can be observed exactly), and thus are not applicable to partial observable environment.

One approach for dealing with such states, is choosing actions based on an entire history of actions and observations. This approach, however, yields extremely high-dimensional inputs to value functions or policies. Thus, learning a low-dimensional representations of such histories is an important challenge in such approaches. Such representations should take information about the task at hand into account, such that non-relevant distractor dimensions can be ignored. Predictive state representations have shown promising results for finding representations in partially-observable problems [Littman et al., 2001, Boots et al., 2011]. So far, however, their application in on-policy learning of challenging continuous robotics task has been limited. I suspect this is due to greedy policy updates that can cause instability, therefore, combining such learned representations with the kind of bounded updates investigated in Chapter 3 would be a promising solution strategy.

In cases where no external rewards are given, but the robot aims at gathering information like in Chapter 2, the robot is also faced with a partially observable state. Thus, methods developed to find optimal control actions in such environments, such as belief-space planning, could be used to efficiently find actions that minimize the long-term posterior entropy, rather than greedily maximizing the one-step look-ahead information gain.

However, minimizing the entropy of the agent's model need not always be desirable. Properties that do not influence the agent's optimal behavior, and are not influenced by it, do not have to be learned in detail. The empowerment objective quantifies the control an agent has over what it is sensing [Jung et al., 2011, Klyubin et al., 2005]. Applying this objective to belief-state could be a promising strategy to drive an agent to belief states where it has maximal control over what it is sensing — i.e., where it is maximally certain what the effects of its actions are.

When using non-parametric kernel methods for reinforcement learning, the kernel defines a prior distribution over objects such as value functions or policies. For most common kernels, this prior only has positive weight over functions that are continuous over the complete input domain. For a wide range of tasks, however, the optimal policy and value function are discontinuous. One way around this problem is to use feature learning techniques that learn this discontinuity, such that the policy or value function is a continuous function of the features. Another topic for future

work is modeling the discontinuous function using a probabilistic prior that allows such discontinuities.

5.2 Outlook

The body of this thesis has focused on making robots learn more autonomous by actively exploring instead of passively waiting for feedback, by using principled techniques instead of manually tweaking heuristics, and by using methods that are robust to real-world, noisy signals by design. The proposed future work aims at scaling such learning algorithms up to more complex tasks and environments by using more prior knowledge, learning subgoals and task-specific features, and exploring more efficiently. If learning methods can indeed be scaled up as proposed, this would yield a powerful approach for robots learning complex tasks with minimal reliance on human design or feedback in large-scale human environments. Thus, such methods could potentially be used to let robots explore and learn how to perform assistive tasks in our everyday environment.

A Publication List

Journal Papers

H. van Hoof, O. Kroemer, and J. Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209, 2014

C. Daniel, H. van Hoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104:337–357, 2016b

H. van Hoof, G. Neumann, and J. Peters. Non-parametric policy search with limited information loss. *Journal of Machine Learning Research*, 2016b. Submitted

H. van Hoof and J. Peters. Generalized exploration in policy search. *Machine Learning*, 2016. Submitted

Conference and Workshop Papers

H. van Hoof, O. Kroemer, H. Ben Amor, and J. Peters. Maximally informative interaction learning for scene exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5152–5158, 2012

H. van Hoof, O. Kroemer, and J. Peters. Probabilistic interactive segmentation for anthropomorphic robots in cluttered environments. In *Proceedings of the International Conference on Humanoid Robots*, pages 169–176, 2013

O. Kroemer, H. van Hoof, G. Neumann, and J. Peters. Learning to predict phases of manipulation tasks as hidden states. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4009–4014, 2014

B. Bischoff, D. Nguyen-Tuong, H. van Hoof, A. McHutchon, C.E. Rasmussen, A. Knoll, J. Peters, and M.P. Deisenroth. Policy search for learning robot control using sparse data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3882–3887, 2014

O. Kroemer, C. Daniel, G Neumann, H. van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510, 2015

H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, pages 995–1003, 2015b

F. F. Veiga, H. van Hoof, J. Peters, and T. Hermans. Stabilizing novel objects by learning to predict tactile slip. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5065–5072, 2015

H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 121–127, 2015a

Z. Yi, R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters. Active tactile object exploration with Gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016. Accepted

H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016a. Accepted

M. Smyk, H. van Hoof, and J. Peters. Learning a multi-task model for control of a compliant robot. In *International Workshop on Cognitive Robotics*, 2016. Accepted

V. Tangkaratt, H. van Hoof, S. Parisi, G. Neumann, J. Peters, and M. Sugiyama. Policy search with high-dimensional context variables. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2016. Submitted

B Curriculum Vitae

Current position

Since 2011	Ph.D. student at Intelligent Autonous Systems Group Technische Universität Darmstadt, Germany Supervisor: Prof. Dr. J. Peters.

Educational Background

2008-2011	*Master of Science in Artificial Intelligence (with honors)* "Autonomous Perceptive Systems" track. University of Groningen, the Netherlands. Thesis: "Interaction between Face Detection and Learning Tracking Systems for Autonomous Robots". Supervised by Dr. T. van der Zant, Dr. M. Wiering, Dr. P. F. Dominey and Prof. Dr. L. Schomaker.
2005-2008	*Bachelor of Science in Artificial Intelligence (with honors)* University of Groningen, the Netherlands. Thesis: "Using Different Methods to Direct a Robot's Attention". Supervised by G. Kootstra and S. de Jong.

Internships

2010	Research internship, Robot Cognition Laboratory, INSERM U846, Lyon, France.

Organized events

R:SS 2016 Workshop on Robot-Environment Interaction for Perception and Manipulation.

Humanoids 2014 Workshop on Active Learning in Robotics.

Invited Talks

2016	RLL and MRL labs, McGill University
2016	CSAIL, Massachusetts Institute of Technology
2016	Alice, Rijksuniversiteit Groningen
2015	Max Planck Institute for Intelligent Systems
2015	Robot Learning Lab, UC Berkeley
2015	RESL and CLMC labs, University of Southern California
2014	Machine Learning and Robotics Lab, Universität Stuttgart
2014	Department of Computing, Imperial College London
2013	Robotics and Biology Group, TU Berlin

Teaching

Teaching assistant, Technische Universität Darmstadt

Technical Foundations of Computer Science	2014-2015
Computational Engineering	2013-2014
Machine Learning Lecture	2013
Robot Learning Lecture	2012-2013
Robot Learning Project	2011-2012, 2012-2013, 2013, 2014, 2014-2015, 2015-2016, 2016

Guest lectures, Technische Universität Darmstadt

Technical Foundations of Computer Science	2014-2015

Lecture assistant, Rijksuniversiteit Groningen

Multi-agent Systems Lecture	2011
Robotics Lab Course	2010-2011
Language and Speech Technology Lab Course	2009
Introduction to Logic Lecture	2008

Reviewing

Journals

2016	IEEE Transactions on Robotics
2016	Autonomous Robots
2016	IEEE Transactions on Automation Science and Engineering (T-ASE)
2014	Journal of Machine Learning Research (JMLR)
2013	Autonomous Robots: Special Issue ‘Beyond Grasping’

Reviewing (continued)

Conferences and workshops

2016 IEEE International Conference on Humanoid Robots (Humanoids)
2016 International Symposium on Experimental Robotics (ISER)
2016 Robotics: Science and Systems (R:SS)
2016 PC, International Joint Conference on Artificial Intelligence (IJCAI)
2015 Robotics: Science and Systems (R:SS)
2014 IEEE International Conference on Robotics and Automation (ICRA) 2015
2014 Neural Information Processing Systems (NIPS)
2014 IEEE International Conference on Intelligent Robots and Systems (IROS)
2013 NIPS WS on Advances in Machine Learing for Sensorimotor Control
2013 IEEE International Conference on Robotics and Automation (ICRA) 2014
2013 Neural Information Processing Systems (NIPS)
2013 International Joint Conference on Artificial Intelligence (IJCAI)
2011 IEEE International Conference on Robotics and Automation (ICRA) 2012

Supervision

Reubold, J. (2014). Master Thesis. (joint supervision with Heni Ben Amor)
3D Object Reconstruction from Partial Views

Schoengen, S. (2013). Bachelor Thesis.
Visual Feature Learning for Interactive Segmentation

Notz, D. (2013). Bachelor Thesis.
Reinforcement Learning for Planning in High-Dimensional Domains

Smyk, M. (2014). Bachelor Thesis.
Learning Generalizable Models for Compliant Robots,

Huhnstock, N. (2014). Bachelor Thesis. (joint supervision with Filipe Veiga)
Tactile Sensing for Manipulation,

Marg, V. (2016). Bachelor Thesis.
Reinforcement Learning for a Dexterous Manipulation Task.

Bibliography

A. Abdolmaleki, R. Lioutikov, J. Peters, N. Lau, L. Reis, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

D. Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII - 1983*, volume 1117, pages 1–198. Springer Berlin / Heidelberg, 1985.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 19–26. AUAI Press, 2009.

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. Data-efficient learning of feedback policies from image pixels using deep dynamical models. Technical Report 1510.02173, ArXiv, 2015.

M. G. Azar, V. Gómez, and B. Kappen. Dynamic policy programming with function approximation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, pages 119–127, 2011.

J.A. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003a.

J.A. Bagnell and J. Schneider. Policy search in reproducing kernel Hilbert space. Technical Report RI-TR-03-45, CMU, 2003b.

A. S. Barreto, D. Precup, and J. Pineau. Reinforcement learning using kernel-based stochastic factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 720–728, 2011.

P. L. Bartlett. An introduction to reinforcement learning theory: Value function methods. In *Advanced Lectures on Machine Learning*, pages 184–202. Springer Berlin Heidelberg, 2003.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

D. Beale, P. Iravani, and P. Hall. Probabilistic models for robot-based object segmentation. *Robotics and Autonomous Systems*, 59(12):1080–1089, 2011.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. ISSN 1935-8237.

N. Bergström, C.H. Ek, M. Björkman, and D. Kragić. Scene understanding through autonomous interactive perception. In *Proceedings of the International Conference on Computer Vision Systems*, pages 153–162, 2011.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific Belmont, MA, 1995.

D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

B. Bischoff, D. Nguyen-Tuong, H. van Hoof, A. McHutchon, C.E. Rasmussen, A. Knoll, J. Peters, and M.P. Deisenroth. Policy search for learning robot control using sparse data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3882–3887, 2014.

J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 2016.

W. Böhmer, S. Grünewälder, Y. Shen, M. Musial, and K. Obermayer. Construction of approximation spaces for reinforcement learning. *The Journal of Machine Learning Research*, 14(1):2067–2118, 2013.

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotics Research*, 30:954–956, 2011.

B. Boots, A. Gretton, and G. J. Gordon. Hilbert space embeddings of predictive state representations. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*, volume 39. CRC press, 2010.

J. Carreira and C. Sminchisescu. CPMC: automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

L. Chang, J.R. Smith, and D. Fox. Interactive singulation of objects from a pile. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

N. Chen, J. Bayer, S. Urban, and P. van der Smagt. Efficient movement representation by embedding dynamic movement primitives in deep autoencoders. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, 2015.

M. Cho, Y. M. Shin, and K. M. Lee. Co-recognition of image pairs by data-driven Monte Carlo image exploration. In *Proceedigns of the European Conference on Computer Vision*, pages 144–157, 2008.

A. Collet, D. Berenson, S.S. Srinivasa, and D. Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 48–55, 2009.

B. C. da Silva, G. Konidaris, and A. G. Barto. Learning parameterized skills. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1679–1686, 2012.

C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 2016a.

C. Daniel, H. van Hoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104:337–357, 2016b.

P. Dayan and G.E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 761–768, 1998.

R. Dearden, N. Friedman, and D. Andre. Model based Bayesian exploration. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 150–159, 1999.

M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7):1508–1524, 2009.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, pages 388–403, 2013.

J. Denzler and C.M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 145–157, 2002.

R. Detry, N. Pugeault, and J.H. Piater. A probabilistic framework for 3D visual object representation. *Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1790–1803, 2009.

T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

F. Doshi-Velez, D. Wingate, N. Roy, and Tenenbaum J. B. Nonparametric Bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 532–540, 2010.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 20, pages 154–161, 2003.

P. Englert and M. Toussaint. Combined optimization and reinforcement learning for manipulation skills. In *Proceedings of the Robotics: Science and Systems Conference (R:SS)*, volume 21, pages 388–403, 2016.

C. Erdogan, M. Paluri, and F. Dellaert. Planar segmentation of RGBD images using fast linear fitting and Markov chain Monte Carlo. In *Proceedings of the Conference on Computer and Robot Vision (CRV)*, pages 32–39, 2012.

M.M. Fard, Y. Grinberg, A.-M. Farahmand, J. Pineau, and D. Precup. Bellman error based feature generation using random projections on sparse spaces. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3030–3038, 2013.

C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Proceedings of the IEEE International Conference on Robotics and Automations (ICRA)*, 2016.

M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

P. Fitzpatrick and G. Metta. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361 (1811):2165–2185, 2003.

P.-E. Forssén. Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1–8, 2007.

E. B. Fowlkes and C.L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2 edition, 2004.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721 –741, 1984.

M. Ghavamzadeh and S. Mahadevan. Hierarchical policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 226–233, 2003.

M. Ghavamzadeh, A. Lazaric, O. Maillard, and R. Munos. LSTD with random projections. In *Advances in Neural Information Processing Systems (NIPS)*, pages 721–729, 2010.

S. Griffith, J. Sinapov, M. Miller, and A. Stoytchev. Toward interactive learning of object categories by a robot: A case study with container and non-container objects. In *Proceedings of the IEEE International Conference on Development and Learning*, pages 1–6. IEEE, 2009.

S. Grünewälder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil. Conditional mean embeddings as regressors. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1823–1830, 2012a.

S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton. Modelling transition dynamics in MDPs with RKHS embeddings. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 535–542, 2012b.

M. Guo, Y. Liu, and J. Malec. A new Q-learning algorithm based on the Metropolis criterion. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2140–2143, 2004.

N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1): 1–18, 2003.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations*, 2016.

K. Hausman, F. Balint-Benczedi, D. Pangercic, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz. Tracking-based interactive segmentation of textureless objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1122–1129, 2013.

K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme. Active articulation model estimation through interactive perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3305–3312, 2015.

E. Herbst, X. Ren, and D. Fox. RGB-D object discovery via multi-scene analysis. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4850–4856, 2011.

T. Hermans, J.M. Rehg, and A. Bobick. Guided pushing for object singulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4783–4790, 2012.

J.M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 918–926, 2014.

S. Höfer and O. Brock. Coupled learning of action parameters and forward models for manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

M. Hoffman, A. Doucet, N. D. Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *Advances in Neural Information Processing Systems (NIPS)*, pages 665–672, 2007.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Robust grasping under object pose uncertainty. *Autonomous Robots*, 31(2):253–268, 2011.

M.F. Huber, T. Dencker, M. Roschani, and J. Beyerer. Bayesian active object recognition via Gaussian process regression. In *Proceedings of the International Conference on Information Fusion*, pages 1718–1725, 2012.

IFR. Executive summary world robotics 2015. Technical report, International Federation of Robotics, 2015.

R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

T. Jung and D. Polani. Kernelizing LSPE(λ). In *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 338–345, 2007.

T. Jung, D. Polani, and P. Stone. Empowerment for continuous agent-environment systems. *Adaptive Behavior*, 19(1):16–39, 2011.

L. P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 167–173, 1993.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

S. A. Kakade. Natural policy gradient. In *Advanced in Neural Information Processing Systems (NIPS)*, 2002.

S. A. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, volume 2, pages 267–274, 2002.

D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 272–277, 2008.

D. Katz, Y. Pyuro, and O. Brock. Learning to manipulate articulated objects in unstructured environments using a grounded representation. In *Proceedings of the Robotics: Science and Systems Conference (R:SS)*, pages 254–261, 2008.

D. Katz, A. Orthey, and O. Brock. Interactive perception of articulated objects. In *International Symposium of Experimental Robotics*, 2010.

J. Kenney, T. Buckley, and O. Brock. Interactive segmentation for manipulation in unstructured environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1377–1382, 2009.

D. I. Kim and G. S. Sukhatme. Interactive affordance map building for a robotic task. In *Intelligent Robots and Systems, International Conference on*, pages 4581–4586, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

A. S. Klyubin, D. Polani, and C. L. Nehaniv. All else being equal be empowered. In *Proceedings of the European Conference on Artificial Life*, pages 744–753, 2005.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856, 2009.

J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2650–2655, 2011.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 11(32):1238–1274, 2013.

N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2619–2624, 2004.

G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1015–1023, 2009.

G. D. Konidaris, Osentoski S., and Thomas P. S. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 380–385, 2011.

P. Kormushev and D. G. Caldwell. Direct policy search reinforcement learning based on particle filtering. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL)*, 2012.

J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Annual Conference on Genetic and Evolutionary Computation*, pages 1061–1068, 2013.

D. Kraft, R. Detry, N. Pugealt, E. Başeski, F. Guerin, J.H. Piater, and N. Krüger. Development of object and grasping knowledge by robot exploration. *IEEE Transactions on Autonomous Mental Development*, 2(4):368–383, 2010.

M. Krainin, B. Curless, and D. Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5031–5037, 2011a.

M. Krainin, P. Henry, X. Ren, and D. Fox. Manipulator and object tracking for in-hand 3D object modeling. *The International Journal of Robotics Research*, 30(11):1311–1327, 2011b.

O. Kroemer and J. Peters. A non-parametric approach to dynamic programming. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1719–1727, 2011.

O. Kroemer, H. van Hoof, G. Neumann, and J. Peters. Learning to predict phases of manipulation tasks as hidden states. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4009–4014, 2014.

O. Kroemer, C. Daniel, G Neumann, H. van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510, 2015.

J. Kulick, R. Lieck, and M. Toussaint. Active learning of hyperparameters: An expected cross entropy criterion for active model selection. Technical Report 1409.7552, arXiv, 2014.

J. Kulick, S. Otte, and M. Toussaint. Active exploration of joint dependency structures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2598–2604, 2015.

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 National Conference on Artificial Intelligence (AAAI)*, 2013.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4: 1107–1149, December 2003.

S. Lange, M. Riedmiller, and A. Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks*, 2012.

G. Lever and R. Stafford. Modelling policies in MDPs in reproducing kernel Hilbert space. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, pages 590–598, 2015.

S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1071–1079, 2014.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

W. H. Li and L. Kleeman. Autonomous segmentation of near-symmetric objects through vision and robotic nudging. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3604–3609, 2008.

W. H. Li and L. Kleeman. Interactive learning of visually symmetric objects. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4751–4756, 2009.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. Technical Report 1509.02971, arXiv, 2015.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2016.

L.-J. Lin. *Reinforcement Learning for Robots using Neural Networks*. PhD thesis, Carnegie Mellon University, 1993.

R. Lioutikov, A. Paraschos, G. Neumann, and J. Peters. Sample-based information-theoretic stochastic optimal control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

D. Little and F. Sommer. Learning and exploration in action-perception loops. *Frontiers in Neural Circuits*, 7:37, 2013.

M. L. Littman, R. S. Sutton, and S. P. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 1555–1561, 2001.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Z. Lu, D. Guo, A. Bagheri Garakani, K. Liu, A. May, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, and F. Sha. A comparison between deep neural nets and kernel acoustic models for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 2016.

J. Macedo, C. Santos, and L. Costa. Using cost-regularized kernel regression with a high number of samples. In *Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 199–204, 2014.

S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Biases and variance in value function estimates. *Management Science*, 53(2):308–322, February 2007.

R. Martín-Martín and O. Brock. Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2494–2501, 2014.

J. Mattner, S. Lange, and M. Riedmiller. Learn to swing up and balance a real pole based on raw visual input data. In *International Conference on Neural Information Processing*, pages 126–133, 2012.

A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (10):1584–1601, 2006.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

S. A. Mobin, J. A. Arnemann, and F. Sommer. Information-based learning by agents in unbounded state spaces. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3023–3031, 2014.

J. Modayil and B. Kuipers. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, 56(11):879–890, 2008.

J. Morimoto and K. Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1):37–51, 2001.

R. Munos. Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791, 2006.

H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

L. Natale, F. Orabona, G. Metta, and G. Sandini. Exploring the world through grasping: a developmental approach. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 559–565, 2005.

A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

Y. Nishiyama, A. Boularias, A. Gretton, and K. Fukumizu. Hilbert space embeddings of POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 644–653, 2012.

D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

P. A. Ortega and D. A. Braun. A minimum relative entropy principle for learning and acting. *Journal of Artificial Intelligence Research*, 38:475–511, 2010.

I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2377–2386, 2016.

S. Otte, J. Kulick, M. Toussaint, and O. Brock. Entropy-based strategies for physical exploration of the environment's degrees of freedom. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 615–622, 2014.

J. Pajarinen and V. Kyrki. Decision making under uncertain segmentations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1303–1309, 2015.

R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems (NIPS)*, pages 1043–1049, 1998.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 752–759, 2008.

J. Pazis and R. Parr. Non-parametric approximate linear programming for MDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 459–464, 2011.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2219–2225, 2006.

J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 745–750, 2007.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, pages 682–97, 2008a.

J. Peters and S. Schaal. Natural actor critic. *Neurocomputing*, 71(7-9):1180–1190, 2008b.

J. Peters, J. Kober, and D. Nguyen-Tuong. Policy learning – a unified perspective with applications in robotics. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL)*, 2008.

J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI), Physically Grounded AI Track*, pages 1607–1612, 2010.

M. Pirotta, M. Restelli, A. Pecorino, and D. Calandriello. Safe policy iteration. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 307–315, 2013.

A.R. Pope and D.G. Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

D. Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2000.

S. J. Pundlik and S. T. Birchfield. Real-time motion segmentation of sparse feature points at any speed. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(3):731–742, 2008.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.

C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 4, page 1, 2003.

K. Rawlik, M. Toussaint, and S. Vijayakumar. Path integral control by reproducing kernel Hilbert space embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013a.

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3052–3056. AAAI Press, 2013b.

C. Rother, V. Kolmogorov, T. Minka, and A. Blake. Cosegmentation of image pairs by histogram matching – incorporating a global constraint into MRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 993–1000, 2006.

E. A. Rückert, G. Neumann, M. Toussaint, and W. Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience*, 6: 97, 2013.

T. Rückstieß, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1):14–24, 2010.

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6): 233–242, 1999.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research.*, pages 561–572, 2005.

D. Schiebener, J. Morimoto, T. Asfour, and A. Ude. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior*, 21(5):328–345, 2013.

D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4959–4966, 2014.

A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 243–248, 2009.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5): 1000–1017, 1999.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer Berlin Heidelberg, 2001.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

M. Seeger, C.K.I. Williams, and N.D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on Artificial Intelligence and Statistics*, 2003.

F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

J. Sinapov, T. Bergquist, C. Schenck, U. Ohiri, S. Griffith, and A. Stoytchev. Interactive object recognition using proprioceptive and auditory feedback. *The International Journal of Robotics Research*, 30(10): 1250–1262, 2011.

S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, 1992.

M. Smyk, H. van Hoof, and J. Peters. Learning a multi-task model for control of a compliant robot. In *International Workshop on Cognitive Robotics*, 2016. Accepted.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2006.

L. Song, K. Fukumizu, and A. Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *Signal Processing Magazine, IEEE*, 30 (4):98–111, 2013.

S. Still and D. Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

M. Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 943–950, 2000.

J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3D laser point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2131–2136, 2010.

F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 231–238, 2011.

F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

O. O. Sushkov and C. Sammut. Feature segmentation for object recognition using robot manipulation. In *Proceedings of the Australian Conference on Robotics and Automation*, 2011.

O. O. Sushkov and C. Sammut. Active robot learning of object properties. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2621–2628, 2012.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 216–224, 1990.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063, 1999a.

R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999b.

C. Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

V. Tangkaratt, H. van Hoof, S. Parisi, G. Neumann, J. Peters, and M. Sugiyama. Policy search with high-dimensional context variables. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2016. Submitted.

G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1017–1024, 2009.

E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

N. Tishby and D. Polani. Information theory of decisions and actions. In *Perception-Action Cycle*, pages 601–636. Springer, 2011.

A. Ude, D. Omrčen, and G. Cheng. Making object learning and recognition an active process. *International Journal of Humanoid Robotics*, 5(2):267–286, 2008.

H. van Hoof and J. Peters. Generalized exploration in policy search. *Machine Learning*, 2016. Submitted.

H. van Hoof, O. Kroemer, H. Ben Amor, and J. Peters. Maximally informative interaction learning for scene exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5152–5158, 2012.

H. van Hoof, O. Kroemer, and J. Peters. Probabilistic interactive segmentation for anthropomorphic robots in cluttered environments. In *Proceedings of the International Conference on Humanoid Robots*, pages 169–176, 2013.

H. van Hoof, O. Kroemer, and J. Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209, 2014.

H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 121–127, 2015a.

H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, pages 995–1003, 2015b.

H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016a. Accepted.

H. van Hoof, G. Neumann, and J. Peters. Non-parametric policy search with limited information loss. *Journal of Machine Learning Research*, 2016b. Submitted.

F. F. Veiga, H. van Hoof, J. Peters, and T. Hermans. Stabilizing novel objects by learning to predict tactile slip. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5065–5072, 2015.

N. A. Vien and M. Toussaint. Touch based POMDP manipulation via sequential submodular optimization. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 407–413, 2015.

N.A. Vien, P. Englert, and M. Toussaint. Policy search in reproducing kernel Hilbert space. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008.

R. Vuga, B. Nemec, and A. Ude. Enhanced policy adaptation through directed explorative learning. *International Journal of Humanoid Robotics*, 12(03), 2015.

N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. Technical Report 1502.02251, ArXiv, 2015.

C. Watkins and Y. Buttkewitz. Sex as Gibbs sampling: a probability model of evolution. Technical Report 1402.2704, ArXiv, 2014.

M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2728–2736, 2015.

P. Wawrzyński. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5:91–95, 2015.

M. Wiering and M. Otterlo. *Reinforcement learning: State-of-the-art*. Springer Berline Heidelberg, 2012.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum. Bayesian policy search with policy priors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

C. Wirth, J. Fürnkranz, and G. Neumann. Model-free preference-based reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2015.

J. Wyatt. *Exploration and Inference in Learning from Reinforcement*. PhD thesis, University of Edinburgh, College of Science and Engineering, School of Informatics, 1998.

K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Transactions on Graphics*, 34(6):177, 2015.

X. Xu, D. Hu, and X. Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, 2007.

X. Xu, C. Lian, L. Zuo, and H. He. Kernel-based approximate dynamic programming for real-time online learning control: An experimental study. *IEEE Transactions on Control Systems Technology*, 22 (1):146–156, 2014.

Z. Yi, R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters. Active tactile object exploration with Gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016. Accepted.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

A. Zimin and G. Neu. Online learning in episodic Markovian decision processes by relative entropy policy search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1583–1591, 2013.