Article

The International Journal of Robotics Research
2018, Vol. 37(1) 137–154

DOI: 10.1177/0278364917743795
journals.sagepub.com/home/ijr

Learning manipulation skills from a single demonstration

Peter Englert and Marc Toussaint

Abstract

We consider the scenario where a robot is demonstrated a manipulation skill once and should then use only a few trials on its own to learn to reproduce, optimize, and generalize that same skill. A manipulation skill is generally a high-dimensional policy. To achieve the desired sample efficiency, we need to exploit the inherent structure in this problem. With our approach, we propose to decompose the problem into analytically known objectives, such as motion smoothness, and black-box objectives, such as trial success or reward, depending on the interaction with the environment. The decomposition allows us to leverage and combine (i) constrained optimization methods to address analytic objectives, (ii) constrained Bayesian optimization to explore black-box objectives, and (iii) inverse optimal control methods to eventually extract a generalizable skill representation. The algorithm is evaluated on a synthetic benchmark experiment and compared with state-of-the-art learning methods. We also demonstrate the performance on real-robot experiments with a PR2.

Keywords

Combined optimization and learning, reinforcement learning, imitation learning, manipulation skills

1. Introduction

Manipulation skills share the common goal of controlling *external* degrees of freedom of the environment into a desired state. Coding a policy that controls the internal degrees of freedom of a robot for such manipulations is non-trivial. A main issue is that the external degrees of freedom can only be manipulated through contacts, which are difficult to plan since a precise and detailed physical interaction model is often not available. This issue motivates the use of learning methods for manipulation skills that allow robots to learn how to manipulate the unknown environment. In the last decade, many impressive applications of learning methods in robotics could be accomplished (e.g., aerobatic helicopter flight (Abbeel et al., 2007), robot table tennis (Muelling et al., 2013) and quadruped locomotion (Theodorou et al., 2010)). In this work, we investigate how such learning methods can improve manipulation skills to achieve a higher performance and wider generalization abilities.

We propose a combination of optimal control, episodic reinforcement learning, and inverse optimal control techniques to eventually learn a cost-function representation of manipulation skills, starting from a single demonstration. The initial demonstration is a trajectory for a particular skill scenario, which is used as a starting point for different learning methods. We design the learning methods in such a way that they exploit the common structure of manipulation skills. A key element of this structure is that external degrees of freedom are manipulated through contacts. These contacts play an essential role for the success of manipulation skills. We use this structure in our learning methods by defining a low-dimensional projection of the interaction with the environment, which is the part of the skill that is most difficult to model. We use episodic reinforcement learning for the parts of the skill that are defined by the projection. For the remaining parts, we use analytic motion optimization methods and keep the projection fixed. We finally use the data we collect with rollouts in an inverse optimal control method to acquire a higher-level skill representation that generalizes to different scenarios.

1.1. Assumptions and overview

Learning manipulation skills is, in general, a very difficult problem. One reason is that models about the environment are only partially known. The geometric shape of the environment can usually be obtained from different sensors.

Machine Learning & Robotics Lab, University of Stuttgart, Germany

Corresponding author:
Peter Englert, Universitätsstraße 38, 70569 Stuttgart, Germany.
Email: peter.englert@ipvs.uni-stuttgart.de

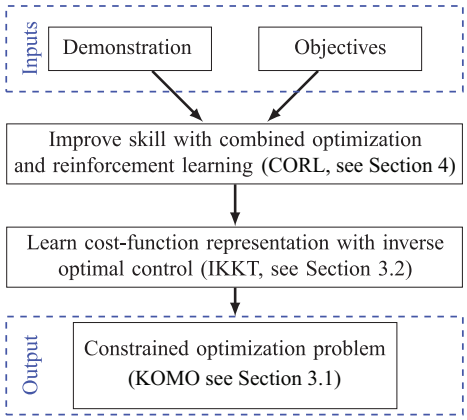


Fig. 1. Overview of our skill learning approach.

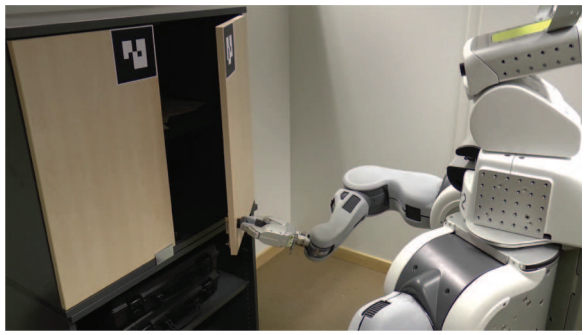


Fig. 2. Cabinet experiment (see Section 6.2).

However, it is more difficult to estimate the precise kinematic structure of the environment and how to manipulate it through contacts. Another reason is that it can be dangerous to apply learning methods when contacts are involved, since they could damage the robot or the environment.

One way to reduce these difficulties is by exploiting the problem structure and by putting prior knowledge into the learning process. We provide a demonstration of the skill as initialization of our method, which is used as a starting point for improving and generalizing the skill. We also restrict our problem class by making the assumption that the environment only consists of rigid bodies that are connected by joints. Another assumption we make is that the success of a skill only depends on a low-dimensional projection of the full motion. We define this projection as the interaction (e.g., contact points, external degrees of freedom) with the environment and use data-efficient and safe learning methods on the low-dimensional projection.

Figure 1 shows an overview of our approach. The inputs are a demonstration and different objectives that should be optimized. Our goal is to get an output in form of a constrained optimization problem that can be optimized to generate motions for different skill scenarios. We use k-order Markov optimization to represent the skill in the form of a cost function and constraints that are defined in environment-dependent feature spaces. We use two learning methods to acquire this abstract skill representation.

In the first part, we propose the structured reinforcement learning method CORL (combined optimization and reinforcement learning) to improve the skill. This method allows us to use analytic and black-box objectives and improves them in a safe and data-efficient manner. To do this, CORL uses a low-dimensional projection of the skill that parametrizes an equality constraint. The skill improvement is done with a combination of analytic optimization on the full motion and episodic reinforcement learning on the low-dimensional projection. During this learning method, we interact with the system and collect data of the performance of different motions.

In the second part, we use the data to learn a cost-function representation of the skill with inverse optimal control. We use the inverse Karush–Kuhn–Tucker (IKKT) method, which allows us to extract the cost function of a constrained optimization problem. It uses the Karush–Kuhn–Tucker (KKT) conditions of the optimization problem to learn the cost function such that the demonstrations fulfill these conditions. We use a set of generic features and constraints that allows us to generalize the skill with respect to different environments. The generalization abilities we want to achieve are different initial states of the robot and different final states of the external degrees of freedom of the environment.

A concrete skill that we consider in the experimental section is to open a cabinet using a PR2 robot (see Figure 2). We initialize the learning with a single demonstration via kinesthetic teaching. We then use a combination of optimal control and reinforcement learning to improve the skill with respect to smoothness and force efficiency. The low-dimensional projection is here defined as the force applied during the opening and the rotation angle of the door knob. Afterwards, we use the acquired trajectory data to learn a cost function that allows the robot to generalize the skill to different initial robot states (see Figure 3(d)) and desired door states (see Figure 3(e))

1.2. Contributions and structure

The goal of our skill learning approach is to find a policy that has a high performance and generalizes to a wide range of different scenarios. The main contributions of this paper are:

(a) A structured learning method, CORL, that combines analytic optimization and episodic reinforcement learning;
(b) Defining a low-dimensional projection for the interaction parts of manipulation skills that allows us to use safe and data-efficient algorithms;
(c) Learning a skill from a single demonstration by bootstrapping it with CORL and generalizing it with inverse optimal control.

This work extends previous work of ours on robot skill learning. Previously (Englert and Toussaint, 2016), we proposed CORL. In this work, we modify the algorithm by merging the two separate policy-improvement parts into a more efficient hierarchical variant. Previously, each part had its own learning loop, which was executed until convergence before continuing with the next part. We modify this to form a single learning loop that combines both policy improvements. This reduces the number of hyperparameters and interactions with the system. This work also integrates prior work on inverse optimal control (Englert and Toussaint, 2015). Instead of using it directly on demonstration data, we integrate this method in a skill learning algorithm, which we use on data collected with a reinforcement learning method.

The structure of this paper is as follows. In Section 2, we present related work in the area of skill learning in robotics. Afterwards, we present in Section 3 a background on trajectory optimization and inverse optimal control. In Section 4, we present CORL. In Section 5, we combine the different parts into an algorithm that allows us to learn manipulation skills from a single demonstration. In Section 6, we evaluate our approach on different synthetic and real-robot problems and compare it with state-of-the-art learning methods.

2. Related work

In reinforcement learning, an agent learns a policy by interacting with its environment (Sutton and Barto, 1998). In this section, we will cover related work on different learning approaches with a special focus on robot manipulation skills.

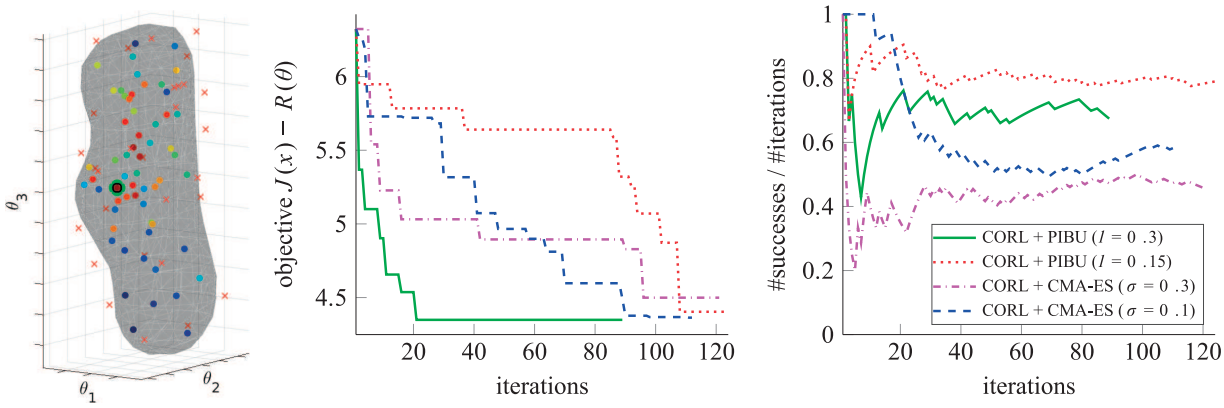
2.1. Learning manipulation skills

Policy search is a widely used technique to learn skills in robotics (Kober et al., 2013). One approach, proposed by Kober and Peters (2008), uses dynamic movement primitives as policy representation and the policy search method PoWER to learn the shape and properties of the motion. Another approach, proposed by Kalakrishnan et al. (2011), involves learning force control policies for manipulations. This policy is initialized with position control via imitation and afterwards augmented with a force profile that is learned using the Policy Improvement with Path Integrals (PI^2) reinforcement learning method (Theodorou et al., 2010). Kalakrishnan et al. (2011) use a single reward function that combines different terms (e.g., smoothness, force, tracking errors). The difference in our approach is that we perform learning on two policy parametrizations. This allows us to use efficient Gauss–Newton optimization routines for those parts of a motion where an analytic cost function is available. Further, we combine these routines with an inverse optimal control method that extracts a cost-function representation with higher generalization abilities than dynamic movement primitives. In our experiments, we compare our methods with covariance matrix adaptation, which has been shown to be closely related to PI^2 (see Stulp and Sigaud, 2013a). Levine et al. (2016) proposed to learn a deep convolutional neural network that maps raw images directly to motor torques. End-to-end training is done with a guided policy search (Levine and Koltun, 2013) that iterates between reinforcement learning to generate rollouts and supervised learning to train the neural network. Levine et al. (2016) also add a pretraining step for the initial policy and the vision system to reduce the amount of interaction time. Chebotar et al. (2017) propose an extension of a guided policy search with the reinforcement learning method PI^2. Fu et al. (2016) present a model-based reinforcement learning approach for manipulation skills, using a neural network to represent the object interaction dynamics. This model is used as prior knowledge for new tasks and adapted during learning. Fu et al. (2016) evaluate the approach for different inserting, stacking, and assembling manipulations, and show that only a small amount of data is required. Instead of using a neural network as policy parametrization, we propose to use a high-level constrained optimization problem that generalizes to different skill scenarios. Further, we incorporate prior knowledge in the form of a low-dimensional projection to achieve a directed and sample-efficient learning behavior.

2.2. Episodic reinforcement learning as black-box optimization

A restricted case of episodic reinforcement learning is where only the total return of an episode is observed but not the individual rewards at each time step (Stulp et al., 2013b). This property transforms the problem into a black-box optimization problem that allows one to use standard black-box optimization methods (e.g., covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 2001) or Bayesian optimization (Mockus et al., 1978)). These methods have previously been used to learn parameters in robotics. For example, Bayesian optimization was used to learn gait parameters for locomotion skills (Calandra et al., 2015; Lizotte et al., 2007). Our approach combines this type of reinforcement learning with non-linear optimal control based on motion optimization. This allows us to use the black-box optimization methods only on a low-dimensional projection of the full policy, which leads to faster convergence.

There exist different methods of including additional constraints in Bayesian optimization (Gardner et al., 2014; Gelbart et al., 2014; Gramacy and Lee, 2011; Schonlau et al., 1998). We include a binary success constraint in our formulation that measures whether a rollout was successful. We use this constraint to achieve a secure learning process, avoiding sampling points that strongly violate the constraint. To obtain this, we propose a novel acquisition function that uses the variance of the constraint to guide the exploration on the decision boundary.



(a) Projection space. **(b)** Best candidate. **(c)** Success rate.

(d) Generalization to different initial positions. **(e)** Generalization to different final door angles.

Fig. 3. Cabinet experiment (see Section 6.2). (a) Classification boundary in projection space $\boldsymbol{\theta}$, which classifies the successful parameters (dots) from the failures (red crosses). (b) Best candidate over iterations. (c) Success rate over iterations; this measures how many failures were executed on the system compared with the number of iterations so far. (d, e) Generalization of constrained optimization problem to different initial positions and final door angles.
CMA-ES: covariance matrix adaptation evolution strategy; CORL: combined optimization and reinforcement learning; PIBU: probability of improvement with boundary uncertainty.

2.3. *Safe learning*

An important aspect in learning manipulation skills is safety; that the robot does not damage itself or the environment. Schreiter et al. (2015) propose a safe exploration strategy for a similar problem to ours. They optimize a function in a safe manner where the feasible region is unknown. To do this, they assume that a safety measure is observed when samples are close to the boundary. This information is integrated into a differential entropy exploration criterion to select subsequent candidates and also provides an upper bound for the probability of failure. This approach would most probably lead to fewer failures during the exploration, but it requires additional information about the distance to the decision boundary in critical regions, which is not available in our problem formulation. We compare our approach to this strategy for a synthetic problem in Section 6.1.

Another approach for a safe exploration is proposed by Sui et al. (2015). The SAFEOPT strategy optimizes an unknown function with Bayesian optimization that is combined with a safety criterion of the form that the function value should exceed a certain threshold. Sui et al. (2015) use the concept of reachability to categorize the search space in different sets for safe exploration and exploitation. The next data point is selected by sampling the most uncertain decision. This approach by Berkenkamp and Schoellig (2015) is used to learn a stabilization task on a quadrotor vehicle.

Garcia Polo and Fernandez-Rebollo (2011) introduce a safe reinforcement learning approach that improves demonstrated behavior in a risk-sensitive manner. The behavior is represented with case-based reasoning techniques. The safety criterion is defined with the distance to the nearest neighbor that is limited with a threshold. The exploration involves adding Gaussian noise to the current optimal actions. This approach uses case-based reasoning techniques, which allows the use of multiple trajectories as demonstration.

Achiam et al. (2017) propose constrained policy optimization that uses constrained Markov decision processes to achieve safe learning. They derive a bound on the difference between the rewards of two different policies; this is used to update a policy while guaranteeing improvement on the return and to satisfy a constraint. Achiam et al. (2017) show that their method can be used to train high-dimensional neural network policies for robotics tasks.

None of these methods for safe or Bayesian exploration would be sample-efficient when directly applied for the high-dimensional non-stationary policy. However, they

could be used within our CORL framework, as demonstrated for GP-UCB and safe active learning (SAL) in the evaluations.

2.4. Combined optimization and learning

There exist a number of approaches that combine learning and optimization methods. The advantage of this combination is that models can be used in the optimization problems. This usually leads to a lower-dimensional space for the learning part, which results in fewer rollouts until convergence. Rückert et al. (2013) introduced a reinforcement learning algorithm for planning movement primitives that uses a two-layered learning formulation. In an outer loop, the CMA-ES policy search method optimizes an extrinsic cost function that measures the task performance. In this policy, a search is made over parameters that are used in the inner loop to define a cost function for a trajectory optimization problem. This problem is used to compute trajectories that are fed back as input to the extrinsic cost function. A core difference from our approach is that Rückert et al. (2013) directly couple the objective functions with each other in a hierarchical way and only optimize the extrinsic objective function. The intrinsic objective function is only used to perform rollouts. In our formulation, we optimize both objectives. Additionally, we use a safety constraint to guide the learning in a secure manner.

Kupcsik et al. (2013) proposed a policy search method that combines model-free reinforcement learning with learned forward models. They learn probabilistic forward models of the robot and the skill, which are used to generate artificial samples in simulation. These samples are combined with real-world rollouts to update the policy. The relative entropy policy search method (Peters et al., 2010) is used to maximize the reward and balance the exploration and experience loss by remaining close to the observed data. A main difference from our approach is that we divide the problem into model-based motion optimization, which improves the motion efficiently, and reinforcement learning, which improves the skill by exploring a low-dimension representation. A further difference is that Kupcsik et al. (2013) learn a model of the task that is used in internal simulations, whereas we directly learn a model that maps parameters to return.

Vuga et al. (2015) introduced an approach that combines the PI^2 reinforcement learning method with the optimization algorithm iterative learning control (Bristow et al., 2006). The policy is represented with a dynamic movement primitive. This approach uses iterative learning control as an exploration strategy in the first part of the learning. In the second part, random exploration is used to fine-tune the policy. Vuga et al. (2015) use iterative learning control to adapt the trajectory and speed profile. Our approach differs, especially with respect to the two policy parametrizations, which allow us to use the reinforcement learning method in a secure and data-efficient manner with Bayesian optimization. Our analytic optimization method also allows us to define cost functions in arbitrary feature spaces.

2.5. Inverse optimal control

Inverse optimal control is used to extract a cost function from data (Levine et al., 2011; Ng and Russell, 2000; Ziebart et al., 2008). Many successful applications in different areas have demonstrated the capabilities of this idea, including the learning of quadruped locomotion (Kolter et al., 2008), helicopter acrobatics (Abbeel et al., 2010) and simulated car driving (Abbeel and Ng, 2004; Levine and Koltun, 2012). For a broader overview on inverse optimal control approaches, we refer the reader to the survey paper of Zhifei and Joo (2012) and for an overview on imitation learning in robotics we recommend Argall et al. (2009). Kalakrishnan et al. (2013) use inverse optimal control on manipulation skills. They introduce an inverse formulation of the PI^2 reinforcement learning method. The cost function consists of a control cost and a general state-dependent cost term at each time step. Kalakrishnan et al. (2013) maximize the trajectory likelihood for all demonstrations by sampling trajectories around the demonstrations. The method is evaluated on grasping tasks.

Finn et al. (2016) propose the use of a neural network to represent the cost function. This allows one to use nonlinear cost functions and does not require the definition of features by hand. The network is trained in an inner loop of a policy search method and evaluated for placement and pouring tasks using a real robot. In our approach, we focus on learning a cost function of a constrained optimization problem similar to the approach of Puydupin-Jamin et al. (2012), where we also use the KKT condition to define the inverse problem of a constrained optimization problem. We do not directly apply inverse optimal control on the input demonstrations. Instead, we first apply a reinforcement learning method to improve the demonstrations before we extract a cost function.

3. Background on trajectory optimization and inverse optimal control

In the following section, we describe background on how to optimize a trajectory with respect to a cost function and constraints. Afterwards, we describe the inverse problem on how to extract a cost function from trajectories.

3.1. k-order Markov optimization

A trajectory $\bar{\boldsymbol{x}}$ is a sequence of $T+1$ robot configurations $\boldsymbol{x}_t \in \mathbb{R}^Q$ that lead to a total number of $n = Q(T+1)$ parameters. The goal of trajectory optimization is to find a trajectory $\bar{\boldsymbol{x}}$, given an initial configuration $\boldsymbol{x}_0$, that minimizes a certain objective function. In k-order Markov optimization

(Toussaint, 2017), the objective function is defined as

$$f(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{w}) = \sum_{t=1}^{T} \boldsymbol{w}_t^\top \boldsymbol{\phi}_t^2(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) \tag{1}$$

$$= \boldsymbol{w}^\top \Phi^2(\bar{\boldsymbol{x}}, \boldsymbol{y}) \tag{2}$$

This defines the objective as a weighted sum over all time steps, where the costs are defined in the form of squared features $\boldsymbol{\phi}$. Each cost term depends on a k-order tuple of consecutive configurations $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$, containing the current and k previous robot configurations. In addition to the robot configuration $\tilde{\boldsymbol{x}}_t$, we use external parameters of the environment $\boldsymbol{y}$ to contain information that is important for planning the motion (e.g., object positions or goal states). In addition to the task costs, we also consider inequality and equality constraints

$$\forall_t \qquad \boldsymbol{g}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) \leq \boldsymbol{0}, \qquad \boldsymbol{h}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) = \boldsymbol{0} \tag{3}$$

which, as features $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$, can refer to arbitrary task spaces. The resulting optimization problem is

$$\begin{aligned} \bar{\boldsymbol{x}}^\star = \arg\min_{\bar{\boldsymbol{x}}} & f(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{w}) \\ \text{s.t.} \quad & \boldsymbol{g}(\bar{\boldsymbol{x}}, \boldsymbol{y}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}) = \boldsymbol{0} \end{aligned} \tag{4}$$

where $\boldsymbol{g}$ and $\boldsymbol{h}$ are vector functions that contain all inequality and equality constraints. The equality constraints are, in our approach, mostly used to represent persistent contacts with the environment (e.g., $\boldsymbol{h}$ describes the distance between hand and object that should be *exactly* 0). The motivation for using equality constraints for contacts, instead of using cost terms in the objective function as in equation (1), is the fact that minimizing costs does not guarantee that they will become 0, which is essential for establishing a contact. We incorporate the constraints with the augmented Lagrangian method and solve the resulting problem with Gauss–Newton optimization (Wright and Nocedal, 1999). Thereby, we exploit the structure of the gradient and Hessian for efficient optimization (see Toussaint, 2017, for more details). In addition to the solution $\bar{\boldsymbol{x}}^\star$, we also get the Lagrange parameters $\boldsymbol{\lambda}^\star$, which provide information on when the constraints are active during the motion. This knowledge can be used to make the control of interactions with the environment more robust (see Toussaint et al., 2014).

In this paper, we use the problem representation in equation (4) as the output of our skill learning algorithm with the goal of generalizing to a wide range of environments $\boldsymbol{y}$. We also use k-order Markov optimization within both learning steps of our algorithm for improving and generalizing the skill with respect to analytic cost functions.

3.2. Inverse Karush–Kuhn–Tucker (IKKT) method

In this section, we describe the IKKT inverse optimal control method (Englert and Toussaint, 2015). The core idea is to learn a cost function from data of the form $D = \{\bar{\boldsymbol{x}}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{p}$. We make the assumption that D is optimal and we aim to learn the weight vector $\boldsymbol{w}$ of equation (4) in such a way that the KKT optimality conditions are fulfilled for D.

The inverse optimal control objective is derived from the Lagrange function of the problem in equation (4)

$$L(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = f(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{w}) + \boldsymbol{\lambda}^\top \begin{bmatrix} \boldsymbol{g}(\bar{\boldsymbol{x}}, \boldsymbol{y}) \\ \boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}) \end{bmatrix} \tag{5}$$

and the KKT conditions. The first KKT condition states that for an optimal solution $\bar{\boldsymbol{x}}^\star$ the condition $\nabla_{\bar{\boldsymbol{x}}} L(\bar{\boldsymbol{x}}^\star, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = \boldsymbol{0}$ must be fulfilled. With the gradient of equation (1)

$$\nabla_{\bar{\boldsymbol{x}}} f(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{w}) = 2 \boldsymbol{J}_\phi(\bar{\boldsymbol{x}}, \boldsymbol{y})^\top \operatorname{diag}(\boldsymbol{w})\, \boldsymbol{\Phi}(\bar{\boldsymbol{x}}, \boldsymbol{y}) \tag{6}$$

this leads to

$$2 \boldsymbol{J}_\phi(\bar{\boldsymbol{x}}, \boldsymbol{y})^\top \operatorname{diag}(\boldsymbol{w})\, \boldsymbol{\Phi}(\bar{\boldsymbol{x}}, \boldsymbol{y}) + \boldsymbol{\lambda}^\top \boldsymbol{J}_c(\bar{\boldsymbol{x}}, \boldsymbol{y}) = \boldsymbol{0} \tag{7}$$

where the matrix $\boldsymbol{J}_c$ is the Jacobian of all constraints and $\boldsymbol{J}_\Phi$ is the Jacobian of the features $\boldsymbol{\Phi}$. We assume that the demonstrations are optimal and should fulfill these conditions. Therefore, the inverse optimal control problem can be viewed as searching for a parameter $\boldsymbol{w}$, such that this condition is fulfilled for all demonstrations.

We express this idea in terms of the loss function

$$\ell(\boldsymbol{w}, \boldsymbol{\lambda}) = \sum_{i=1}^{p} \ell^{(i)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(i)}) \tag{8}$$

with

$$\ell^{(i)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(i)}) = \left|\left| \nabla_{\bar{\boldsymbol{x}}} L(\bar{\boldsymbol{x}}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{w}) \right|\right|^2 \tag{9}$$

where we sum over p demonstrations of the scalar product of the first KKT condition. In equation (8), i enumerates the demonstrations and $\boldsymbol{\lambda}^{(i)}$ is the dual to the demonstration $\bar{\boldsymbol{x}}^{(i)}$ under the problem defined by $\boldsymbol{w}$. Note that the dual demonstrations are initially unknown and, of course, depend on the underlying cost function f. More precisely, $\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{(i)}(\bar{\boldsymbol{x}}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{w})$ is a function of the primal demonstration, the environment configuration of that demonstration, and the underlying parameters $\boldsymbol{w}$. And $\ell^{(i)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(i)}(\boldsymbol{w})) = \ell^{(i)}(\boldsymbol{w})$ becomes a function of the parameters only (we think of $\bar{\boldsymbol{x}}^{(i)}$ and $\boldsymbol{y}^{(i)}$ as given, fixed quantities, as in equations (8) and (9)).

Given that we want to minimize $\ell^{(i)}(\boldsymbol{w})$, we can substitute $\boldsymbol{\lambda}^{(i)}(\boldsymbol{w})$ for each demonstration by inserting equation (7) into equation (9) and choosing the dual solution that analytically minimizes $\ell^{(i)}(\boldsymbol{w})$ *subject to* the KKT's complementarity condition

$$\frac{\partial}{\partial \boldsymbol{\lambda}^{(i)}} \ell^{(i)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(i)}) = \boldsymbol{0} \tag{10}$$

$$\Rightarrow \quad \boldsymbol{\lambda}^{(i)}(\boldsymbol{w}) = -2(\tilde{\boldsymbol{J}}_c\, \tilde{\boldsymbol{J}}_c^\top)^{-1} \tilde{\boldsymbol{J}}_c \boldsymbol{J}_\phi^\top \operatorname{diag}(\boldsymbol{\Phi})\, \boldsymbol{w} \tag{11}$$

Note that here the matrix $\tilde{\boldsymbol{J}}_c$ is a subset of the full Jacobian of the constraints $\boldsymbol{J}_c$ that contains only the active constraints during the demonstration, which we can evaluate as $\boldsymbol{g}$ and $\boldsymbol{h}$, which are independent of $\boldsymbol{w}$. This ensures that equation (11) is the minimizer subject to the complementarity condition. The number of active constraints at each time point has a limit. This limit would be exceeded if more degrees of freedom of the system were constrained than were available.

By inserting equation (11) into equation (9), we get

$$\ell^{(i)}(\boldsymbol{w}) = 4\boldsymbol{w}^\top \underbrace{\mathrm{diag}(\boldsymbol{\Phi})\boldsymbol{J}_\phi(\boldsymbol{I} - \tilde{\boldsymbol{J}}_c^\top(\tilde{\boldsymbol{J}}_c\tilde{\boldsymbol{J}}_c^\top)^{-1}\tilde{\boldsymbol{J}}_c)\boldsymbol{J}_\phi^\top \mathrm{diag}(\boldsymbol{\Phi})}_{\boldsymbol{\Lambda}^{(i)}}\boldsymbol{w}$$

which is the inverse optimal control cost per demonstration. Adding up the loss functions for all demonstrations in equation (8) gives the total IKKT loss of

$$\ell(\boldsymbol{w}) = \boldsymbol{w}^\top \Lambda \boldsymbol{w} \qquad \text{with} \quad \Lambda = 4\sum_{i=1}^{p} \Lambda^{(i)} \tag{12}$$

The resulting optimization problem is

$$\begin{aligned} \min_{\boldsymbol{w}} \ & \boldsymbol{w}^\top \Lambda \boldsymbol{w} \\ \text{s.t.} \ & \boldsymbol{w} \geq \boldsymbol{0} \\ & \sum_i \boldsymbol{w}_i = 1 \end{aligned} \tag{13}$$

Note that we constrain the parameters $\boldsymbol{w}$ to be positive. This reflects that we want squared cost features to only positively contribute to the overall cost in equation (1). Additionally, we use another constraint to regularize the problem by requiring that the sum of all weights should be one. The latter constraint avoids the singular solution $\boldsymbol{w} = \boldsymbol{0}$, where zero costs are assigned to all demonstrations. Equation (13) is a (convex) quadratic program, for which there exist efficient solvers. The gradient $\boldsymbol{w}^\top\Lambda$ and Hessian Λ are very structured and sparse, which we exploit in our implementations.

In practice, we usually use parametrizations on $\boldsymbol{w}$. This is useful since in the extreme case, when a different parameter is used for each time step, this leads to a very high-dimensional parameter space (e.g., 10 tasks and 300 time steps lead to 3000 parameters). This space can be reduced by using the same weight parameter over all time steps or by activating a task only at some interesting time points. The simplest variant is to use a linear parametrization $\boldsymbol{w}(\boldsymbol{\rho}) = A\boldsymbol{\rho}$, where $\boldsymbol{\rho}$ are the parameters that the inverse optimal control method learns. This parametrization allows a flexible assignment of one parameter to a number of task costs. Further linear parametrizations are radial basis functions or B-spline basis functions over time t to more compactly describe smoothly varying cost parameters. For such linear parametrizations, the problem in equation (13) remains a quadratic program that can be solved very efficiently. It is also possible to use nonlinear mappings of the form $\boldsymbol{w}(\boldsymbol{\rho}) = \mathcal{A}(\boldsymbol{\rho})$ to learn more complex weight functions (see Englert and Toussaint, 2015, for more details).

In our approach, we use IKKT to extract a cost-function representation of a skill that can generalize to new environments $\boldsymbol{y}$. Instead of directly applying IKKT on demonstration data, we only start with a single demonstration and first apply reinforcement learning to collect data while exploring and improving the skill.

4. Combined optimization and reinforcement learning (CORL)

The CORL algorithm is a structured reinforcement learning formulation that combines optimization and episodic reinforcement learning. It starts with a single demonstration and improves the skill with respect to different objective functions. The main idea of the algorithm is to use the benefits of a transition model and analytic cost function when they are available and the flexibility of black-box objectives otherwise. We specifically aim to deal with cases where the policy parameters are high-dimensional ($n \geq 1000$) but at the same time we aim for efficient skill learning from only few (< 100) real-world rollouts. Clearly, for this to be a well-posed problem we need to assume a certain structure in the problem.

4.1. Problem formulation

Our problem formulation consists of an analytically known cost function

$$J : \mathbb{R}^n \to \mathbb{R} \tag{14}$$

a q-dimensional equality constraint

$$\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \boldsymbol{0} \tag{15}$$

that ties every policy parameter $\bar{\boldsymbol{x}}$ and environment configuration $\boldsymbol{y}$ to a low-dimensional projection $\boldsymbol{\theta} \in \mathbb{R}^m$ (details are given later), a black-box return function

$$R : \mathbb{R}^m \to \mathbb{R} \tag{16}$$

and a black-box success constraint

$$S : \mathbb{R}^m \to \{0, 1\} \tag{17}$$

With these four ingredients, we define the generalized reinforcement learning problem

$$\begin{aligned} \min_{\bar{\boldsymbol{x}}, \boldsymbol{\theta}} \ & J(\bar{\boldsymbol{x}}) - R(\boldsymbol{\theta}) \\ \text{s.t.} \ & \boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \boldsymbol{0} \\ & S(\boldsymbol{\theta}) = 1 \end{aligned} \tag{18}$$

That is, we want the best policy parameters $(\bar{\boldsymbol{x}}^\star, \boldsymbol{\theta}^\star)$ (measured with $J(\bar{\boldsymbol{x}})$ and $R(\boldsymbol{\theta})$) that fulfill a skill (measured with $S(\boldsymbol{\theta}) = 1$). In contrast with the standard optimal control and reinforcement learning problems, which only optimize a single objective function, our formulation splits the objective into an analytic part $J(\bar{\boldsymbol{x}})$, a black-box part $R(\boldsymbol{\theta})$, and a

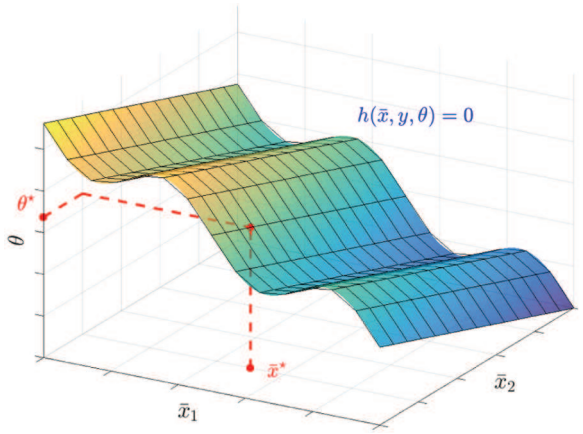


Fig. 4. Projection of two-dimensional $\bar{\boldsymbol{x}}$ to one-dimensional $\boldsymbol{\theta}$ with projection constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \mathbf{0}$.

black-box success constraint $S(\boldsymbol{\theta})$. The analytic cost function $J(\bar{\boldsymbol{x}})$ contains all the costs we know a priori in analytic form. The black-box return function $R(\boldsymbol{\theta})$ and success constraint $S(\boldsymbol{\theta})$ are a-priori unknown and we can only observe noisy samples using rollouts for a given input.

The low-dimensional projection $\boldsymbol{\theta}$ is used to parametrize interactions with the environment (for more details on how we choose $\boldsymbol{h}$ in practice, see Section 5.1). An equality constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta})$ is incorporated to define the relation between the high-dimensional $\bar{\boldsymbol{x}}$ and the environment $\boldsymbol{y}$ to the lower-dimensional $\boldsymbol{\theta}$. We assume that $\boldsymbol{h}$ is smooth and that, for given $\bar{\boldsymbol{x}}$ and $\boldsymbol{y}$, $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \mathbf{0}$ identifies a *unique* $\boldsymbol{\theta}(\bar{\boldsymbol{x}}, \boldsymbol{y}) = \boldsymbol{\theta}$. In this sense, $\boldsymbol{\theta}$ is a projection of $\bar{\boldsymbol{x}}$ and $\boldsymbol{y}$ (see Figure 4 for an example in two dimensions). This projection is formulated in terms of an equality constraint so that, for given $\boldsymbol{\theta}$ and environment $\boldsymbol{y}$, the remaining problem on $\bar{\boldsymbol{x}}$ is a standard constraint optimization problem.

4.2. Approach: Combining optimization with reinforcement learning

We solve the problem in equation (18) by using optimal control methods to improve the policy with respect to the high-dimensional $\bar{\boldsymbol{x}}$ and black-box Bayesian optimization to improve the policy with respect to the low-dimensional $\boldsymbol{\theta}$. We assume that we have an initial policy parametrization $(\bar{\boldsymbol{x}}^0, \boldsymbol{\theta}^0)$ as input to our method that fulfills the skill $(S(\boldsymbol{\theta}^0) = 1)$. A summary of the policy update steps of CORL can be found in the first step of Algorithm 1.

The learning loop of CORL consists of two steps. The first step is black-box Bayesian optimization over $\boldsymbol{\theta}$, which aims at improving $R(\boldsymbol{\theta})$ and fulfilling the constraint $S(\boldsymbol{\theta})$. We define an acquisition function $a(\boldsymbol{\theta})$ in such a way that it explores the parameter space in a secure and data-efficient manner by finding a good tradeoff between making large steps that potentially lead to risky policies and small steps that would require many rollouts. To achieve this goal, we learn a binary classification model of $S(\boldsymbol{\theta})$ to find the boundary between policies that lead to success or failure. This classifier is used to keep the exploration around the feasible region and reduce the number of (negative) interactions with the system. For our domain of manipulation skills, we use the contacts during the manipulations to define a low-dimensional representation $\boldsymbol{\theta}$ (e.g., the contact position). The projection constraints $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta})$ can be computed with robot kinematics that describe the relationship between the full trajectory $\bar{\boldsymbol{x}}$ and environment $\boldsymbol{y}$ to the low-dimensional $\boldsymbol{\theta}$. Optimizing $\boldsymbol{\theta}$ with Bayesian optimization learns which interactions lead to success. For many cases this is reasonable, since the parts of the motion where the robot is performing the contact are difficult to fit into the analytic cost function J and are usually very important in achieving success.

The second step in CORL is constrained optimization and acts on the high-dimensional $\bar{\boldsymbol{x}}$ to improve the analytic cost function $J(\bar{\boldsymbol{x}})$. For this, we use the constrained trajectory optimization framework of Section 3.1. Thereby, the low-dimensional parameter $\boldsymbol{\theta}$ is kept fixed with the equality constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \mathbf{0}$. Fixing the low-dimensional

Algorithm 1

Inputs:
Demonstration: $(\bar{\boldsymbol{x}}^0, \boldsymbol{\theta}^0, \boldsymbol{y})$
Objectives and constraints: $J, R, S, \boldsymbol{h}$

1. Improve skill with CORL (Section 4)
Init $D = (\bar{\boldsymbol{x}}^{(0)}, \boldsymbol{\theta}^{(0)}, R(\bar{\boldsymbol{x}}^{(0)}), J(\boldsymbol{\theta}^{(0)}), S(\boldsymbol{\theta}^{(0)}))$
repeat:
Reinforcement learning of $R(\boldsymbol{\theta})$ and $S(\boldsymbol{\theta})$:
$\boldsymbol{\theta}^{(j)} = \arg\max_{\boldsymbol{\theta}} a(\boldsymbol{\theta}, D)$
Motion optimization for constrained $\boldsymbol{\theta}$:
$\bar{\boldsymbol{x}}^{(j)} = \arg\min_{\bar{\boldsymbol{x}}} J(\bar{\boldsymbol{x}})$ such that $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}^{(j)}) = \mathbf{0}$
Perform rollout with policy parameter $\bar{\boldsymbol{x}}^{(j)}$
Add $(\bar{\boldsymbol{x}}^{(j)}, \boldsymbol{\theta}^{(j)}, J(\bar{\boldsymbol{x}}^{(j)}), R(\boldsymbol{\theta}^{(j)}), S(\boldsymbol{\theta}^{(j)}))$ to D
until no change in policy parameter

2. Learn cost function with IKKT (Section 3.2)
Define a dataset $D^\star$ with the best b candidates of D
Generate features and constraints from $D^\star$
Optimize feature weights

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \ell(\boldsymbol{w})$$
$$\text{such that} \quad \boldsymbol{w} \geq 0, \qquad \sum_i \boldsymbol{w}_i \geq 1$$

Output:
Constrained optimization problem:

$$\min_{\bar{\boldsymbol{x}}} \boldsymbol{w}^{\star\top} \Phi^2(\bar{\boldsymbol{x}}, \boldsymbol{y})$$
$$\text{such that} \quad \boldsymbol{g}(\bar{\boldsymbol{x}}, \boldsymbol{y}) \leq \mathbf{0}, \quad \boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}) = \mathbf{0}$$

parameter $\boldsymbol{\theta}$ means that the resulting policy fulfills a certain property, defined by $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \boldsymbol{0}$. We assume that the success of a skill only depends on $\boldsymbol{\theta}$, which implies that all policy parameters $\bar{\boldsymbol{x}}$ and environments $\boldsymbol{y}$ that fulfill the constraint for a fixed $\boldsymbol{\theta}$ lead to the same outcome.

In the following two sections, first the Bayesian optimization and afterwards the motion optimization are described in detail.

4.3. Reinforcement learning over $\boldsymbol{\theta}$ with unknown success constraints

We introduce an episodic reinforcement learning method to improve the policy with respect to the low-dimensional projection $\boldsymbol{\theta}$. The goal of this improvement strategy is to optimize the black-box return function $R(\boldsymbol{\theta})$ under the success constraint $S(\boldsymbol{\theta})$, so as to have a safe interaction with the system. We use Bayesian optimization to learn a binary classifier for the success constraint $S(\boldsymbol{\theta})$ and a regression model for the return function $R(\boldsymbol{\theta})$. We propose a new acquisition function $a(\boldsymbol{\theta})$ that combines both models in such a way that the next policy is selected in a secure and data-efficient manner. We first introduce the required background on Gaussian processes and Bayesian optimization before introducing our reinforcement learning strategy.

4.3.1. Background on Gaussian processes. For both function approximations, we use Gaussian processes. The advantage of Gaussian processes is that they can express a broad range of different functions and that they provide probability distributions over predictions. A Gaussian process defines a probability distribution over functions (Rasmussen and Williams, 2006). We will first handle the regression and afterwards the classification case.

A Gaussian process is fully specified by a mean function $m(\boldsymbol{\theta})$ and a covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$. In the regression case, we have data of the form $\{\boldsymbol{\theta}_i, r_i\}_{i=1}^d$ with inputs $\boldsymbol{\theta}_i \in \mathbb{R}^m$ and outputs $r_i \in \mathbb{R}$. Predictions for a test input $\boldsymbol{\theta}_\star$ are given by mean and variance

$$\mu(\boldsymbol{\theta}_\star) = m(\boldsymbol{\theta}_\star) + \kappa(\boldsymbol{\theta}_\star)^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{r} \tag{19}$$

$$\mathbb{V}(\boldsymbol{\theta}_\star) = k(\boldsymbol{\theta}_\star, \boldsymbol{\theta}_\star) - \kappa(\boldsymbol{\theta}_\star)^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \kappa(\boldsymbol{\theta}_\star) \tag{20}$$

with $\kappa_i(\boldsymbol{\theta}_\star) = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_\star)$, the Gram matrix K with $K_{ij} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, and training inputs $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d]$ with corresponding targets $\boldsymbol{r} = [r_1, \ldots, r_d]^\top$.

In the binary classification case, the outputs are discrete labels $s \in \{0, 1\}$ and we have data of the form $\{\boldsymbol{\theta}_i, s_i\}_{i=1}^d$. Here, we cannot directly use a Gaussian process to model the output. Therefore, the Gaussian process models a discriminative function $g(\boldsymbol{\theta})$, which defines a class probability via the sigmoid function

$$p(s = 1 | \boldsymbol{\theta}) = \sigma(g(\boldsymbol{\theta})) \tag{21}$$

Since this likelihood is non-Gaussian, the exact posterior over g is not a Gaussian process—one instead uses a Laplace approximation (Nickisch and Rasmussen, 2008). For more details regarding Gaussian processes we refer to Rasmussen and Williams (2006).

4.3.2. Background on Bayesian optimization. Bayesian optimization (Mockus et al., 1978) is a strategy to find the maximum of an objective function $R(\boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^m$, where the function $R(\boldsymbol{\theta})$ is not known in closed-form expression and only noisy observations r of the function value can be made at sampled values $\boldsymbol{\theta}$. These samples are collected in a dataset $\{\boldsymbol{\theta}_i, r_i\}_{i=1}^d$ that is used to build a Gaussian process model of R. The next sample point $\boldsymbol{\theta}_{d+1}$ is chosen by maximizing an acquisition function $a(\boldsymbol{\theta})$. There are many different ways to define this acquisition function (Brochu et al., 2010). One widely used acquisition function is the *probability of improvement* (Kushner, 1964), which is defined as

$$\mathrm{PI}(\boldsymbol{\theta}) = P\left(R(\boldsymbol{\theta}) \geq R(\boldsymbol{\theta}^+)\right) = \Phi\left(\frac{\mu(\boldsymbol{\theta}) - R(\boldsymbol{\theta}^+)}{\sqrt{\mathbb{V}(\boldsymbol{\theta})}}\right) \tag{22}$$

$$\text{with} \quad \boldsymbol{\theta}^+ = \underset{\boldsymbol{\theta} \in \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d\}}{\arg\max} \; R(\boldsymbol{\theta})$$

where Φ is the normal cumulative distribution function. We will make use of this probability of improvement in our acquisition function and extend it for an exploration in a safe manner.

4.3.3. Episodic reinforcement learning over $\boldsymbol{\theta}$. We want to improve the skill by optimizing the parameter $\boldsymbol{\theta}$ with respect to $R(\boldsymbol{\theta})$ and fulfilling the constraint $S(\boldsymbol{\theta})$. To do this we collect data of the form $D = \{\boldsymbol{\theta}^{(i)}, r^{(i)}, s^{(i)}\}_{i=1}^d$, where $\boldsymbol{\theta}$ are the parameters, r is the return and s is the skill outcome. The data D are used to select the next sample $\boldsymbol{\theta}^{(d+1)}$. We use a Gaussian process g_R to model the return function $R(\boldsymbol{\theta})$ and a classifier $\sigma(g_S)$ with Gaussian process g_S to model the success function $S(\boldsymbol{\theta})$. The regression Gaussian process contains only data points that are feasible and lead to success. The classification Gaussian process describes the feasible region of all $\boldsymbol{\theta}$ that lead to skill success. This region is incrementally explored with the goal to find the maximum $R(\boldsymbol{\theta})$ that leads to success.

For both Gaussian processes, we use a squared exponential kernel function

$$\mathrm{k}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma_{\mathrm{sf}}^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}')\right) \tag{23}$$

where $\boldsymbol{\Sigma} = \mathrm{diag}([l_1^2, l_2^2, \ldots, l_m^2])$ is a matrix with squared length scales and σ_{sf} is the signal standard deviation. In the regression model g_R, we use a constant prior mean function of 0. For the classification model g_S, we use a constant prior mean function $m(\boldsymbol{\theta}) = c$ to predict the unfeasible class in regions where no data points are yet available. Therefore, we select a constant c smaller than 0 that allows us to keep the exploration close to the region where data points are available.

We use g_R and g_S to define the acquisition function

$$a_{\text{PIBU}}(\boldsymbol{\theta}) = [g_S(\boldsymbol{\theta}) > 0]\ \text{PI}_{g_R}(\boldsymbol{\theta}) + [g_S(\boldsymbol{\theta}) = 0]\ \mathbb{V}_{g_S}(\boldsymbol{\theta}) \quad (24)$$

that combines the probability of improvement with a boundary uncertainty criterion (PIBU). In equation (24), $[\cdot]$ denotes the Iverson brackets. The first term describes the probability of improvement (cf. equation (22)) of g_R in the inner region of the classifier g_S. The second term is the predictive variance of the Gaussian process classifier g_S on the decision boundary. The first term focuses on exploiting improvement inside the feasible region and the second term focuses on exploring safely on the decision boundary. In each iteration of CORL we optimize equation (24) to find the next low-dimensional parameter $\boldsymbol{\theta}$.

4.4. Motion optimization for constrained $\boldsymbol{\theta}$

After a next candidate of the low-dimensional policy parameter $\boldsymbol{\theta}^{(j)}$ is selected, a backprojection to the full policy representation $\bar{\boldsymbol{x}}^{(j)}$ is necessary to perform a rollout on the actual system. We do this backprojection by selecting the $\bar{\boldsymbol{x}}^{(j)}$ that optimizes $J(\bar{\boldsymbol{x}})$ and fulfills the constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}^{(j)}) = \mathbf{0}$. This leads to the optimization problem

$$\bar{\boldsymbol{x}}^{(j)} = \arg\min_{\bar{\boldsymbol{x}}} J(\bar{\boldsymbol{x}}) \quad \text{s.t.} \quad \boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}^{(j)}) = \mathbf{0} \quad (25)$$

We utilize the k-order Markov optimization framework (see Section 3.1) to optimize this problem by defining the analytically known cost function J as a weighted sum of squared features (see equation (1)) and $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}^{(j)}) = \mathbf{0}$ as an equality constraint (see equation (3)) that is parametrized by $\boldsymbol{\theta}^{(j)}$. The resulting policy parameters $(\boldsymbol{\theta}^{(j)}, \bar{\boldsymbol{x}}^{(j)})$ are executed on the real robot and the observed objectives $(R(\bar{\boldsymbol{x}}^{(j)}), J(\boldsymbol{\theta}^{(j)}), S(\boldsymbol{\theta}^{(j)}))$ are added to the dataset. These steps are repeated until there is no change in the policy parameters.

The optimization problem in equation (25) allows to include a wide variety of objectives and constraints that are necessary for a task (e.g., smoothness, collision avoidance). We now define two objectives for the problem in equation (25) that we use in our experiments for manipulation skills.

4.4.1. Optimizing smoothness of unconstrained motion. Our first objective criterion is smoothness in configuration space while fixing the low-dimensional projection. In our experiments, we define configuration space acceleration features

$$\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t) = (\boldsymbol{x}_t - 2\boldsymbol{x}_{t-1} + \boldsymbol{x}_{t-2}) / \Delta_t^2 \quad (26)$$

that contribute to the k-order Markov optimization objective in equation (1) (alternative smoothness criteria, such as jerk or torque, can also be used). We use this feature to select the next policy parameters $\bar{\boldsymbol{x}}^{(j)}$ by minimizing the problem defined in equation (25). This leads to a smoother motion of the unconstrained part of the motion (e.g., the motion toward the contact).

4.4.2. Optimizing the interaction phase profile. The second objective is to achieve a smoother motion also with respect to the time course of the constraints (e.g., when contacts are established). To do this, we additionally optimize the phase of the trajectory and keep the geometry of the trajectory fixed. Thereby, we assume that the trajectory $\bar{\boldsymbol{x}}$ can be evaluated at time t by interpolating it with splines. We optimize the phase profile $p(t) : [0, T] \rightarrow [0, 1]$ of this trajectory with respect to transition costs. To do this, we discretize $p(t)$ in $K + 1$ points $\hat{p} = [p_0, p_1, \ldots, p_K]$ with the boundary conditions $p_0 = 0$ and $p_K = 1$.

Again, we use the squared configuration space accelerations as a smoothness term that results in an overall cost

$$J(\hat{p}) = \sum_{i=0}^{K} \left((\bar{\boldsymbol{x}}(p_i T) - 2\bar{\boldsymbol{x}}(p_{i-1} T) + \bar{\boldsymbol{x}}(p_{i-2} T)) / \Delta_t^2 \right)^2 + (p_i - 2p_{i-1} + p_{i-2})^2 \quad (27)$$

The second term is a cost term directly on the acceleration of the phase variable. The resulting phase profile $\hat{p}^\star$ defines a new trajectory $\bar{\boldsymbol{x}}$ that is executed on the real system.

5. Learning manipulation skills from a single demonstration

Algorithm 1 shows our skill learning approach, which connects the different learning methods presented in the previous sections. The inputs are a demonstration of the skill and the objective functions. Additionally, a low-dimensional projection $\boldsymbol{\theta}$ is defined with the projection constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta})$ (see Section 5.1 for different ways to define $\boldsymbol{h}$ for manipulations).

In the first step of the algorithm, the structured reinforcement learning method CORL (see Section 4) is applied. We initialize the dataset D with the input demonstration $(\bar{\boldsymbol{x}}^{(0)}, \boldsymbol{\theta}^{(0)})$ and its corresponding performance $(J(\bar{\boldsymbol{x}}^{(0)}), R(\boldsymbol{\theta}^{(0)}), S(\boldsymbol{\theta}^{(0)}))$. Afterwards, we iterate the CORL policy update, which first selects a new low-dimensional projection $\boldsymbol{\theta}^{(j)}$ by maximizing equation (24), which is then mapped on the full policy representation $\bar{\boldsymbol{x}}^{(j)}$ by optimizing equation (25). This policy is executed on the real system, and the observed $J(\bar{\boldsymbol{x}}^{(j)})$, $R(\boldsymbol{\theta}^{(j)})$, and $S(\boldsymbol{\theta}^{(j)})$ are added to the dataset D. This procedure is repeated until there is no more change in the policy parameters. The result of CORL provides us with a dataset D that contains all the rollouts with their performance.

In the second step of Algorithm 1, we select the best b successful data points of D to define a new dataset $D^\star$. This dataset $D^\star$ is used to learn a cost function of the skill with the inverse optimal control method IKKT (see Section 3.2). To do this, we use a set of features and constraints that are specific for manipulations skills, including the projection constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \mathbf{0}$ (see details in Section 5.2). Afterwards, we learn the corresponding weights $\boldsymbol{w}$ with the optimization problem in equation (13), such that the

resulting cost function fulfills the KKT conditions of the dataset $D^\star$.

The resulting constrained optimization problem is the output of our method, which allows us a wide range of generalization abilities to intrinsic or extrinsic changes. Extrinsic changes are, in our case, different environments $\boldsymbol{y}$, which are different initial configurations of the environment (e.g., different object positions) or different target states (e.g., different final door angles). The generalizations to intrinsic changes means the ability that a robot can perform a skill in different ways (e.g., a door can be opened with many different contacts). During the learning with CORL we collected a dataset that contains many parameters $\boldsymbol{\theta}$ that allow the skill to be controlled in different ways.

5.1. Low-dimensional projection $\boldsymbol{\theta}$ as interaction parameters in manipulations

In the application domain of robot manipulation skills, we design the low-dimensional projection $\boldsymbol{\theta}$ as interaction parameters with the environment. This follows our assumption that the interactions are the most important parts of the skill and difficult to model. Essentially, our framework assumes that this interaction parameter space is much lower-dimensional than the full robot motion. The projections can be split in two different types: (1) parameters of the contacts with the environment, where $\boldsymbol{\theta}$ should capture essential parameters of the interaction with the objects, e.g., where and how to establish contact and where to release contact; (2) parameters of the degrees of freedom of the environment, for example, how far to rotate a door handle until it unlocks the door joint. In our experiments with the PR2, we show different combinations of these two projection types.

A concrete constraint that we use in the door opening experiment in Section 6.3 defines the contact points on the door handle. If $\boldsymbol{\phi}_{\text{CP}}(\tilde{\boldsymbol{x}}_{t_C})$ gives the forward kinematics of the robot's contact points at the time of contact t_c and $\boldsymbol{\theta}$ is the point where the robot is grasping the door handle, the projection constraint can be defined as

$$\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \boldsymbol{\phi}_{\text{CP}}(\tilde{\boldsymbol{x}}_{t_C}) - \boldsymbol{\theta} \tag{28}$$

This concept is transferable to different manipulation skills where the contact points are crucial for performance and success.

5.2. IKKT features for manipulations

Manipulating external degrees of freedom shares a common structure that we want to extract in a generic set of features and constraints. Our IKKT formulation includes several cost features and hard constraints. In the real-robot experiments we use the following kinds of feature:

- *Transition features.* Represent the smoothness of the motion (e.g., sum of squared acceleration or torques);
- *Position features.* Represent a body position relative to another body (e.g., between robot gripper and door handle);
- *Orientation features.* Represent orientation of a body relative to another body.

These features are defined at different time steps (e.g., before or after contact change) and for different bodies. We define these features relative to the manipulated objects, such that they can be transferred more easily to different scenarios. Concerning the constraints, we always adopt the projection constraint $\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\theta}) = \boldsymbol{0}$ of CORL (e.g., contact points) as an equality constraint into IKKT. Further constraints are:

- Inequality constraints to avoid collisions with the environment;
- Inequality constraints to stay inside the robot joint limits;
- Equality constraints to fix external degrees of freedom that are not being manipulated;
- Equality constraints to ensure that the final state of external degrees of freedom are reached (e.g., final door state).

Equation (13) is used to compute optimal weights $\boldsymbol{w}$. The features that receive a weight larger than 0 are extracted and used in the resulting policy.

6. Experiments

We evaluate our approach on the synthetic optimization problem and multiple robot manipulation experiments. In both cases, we compare the performance with alternative learning methods. We address different manipulations with a PR2, where good models are available for the part of the motion where the robot is moving freely but it is hard to obtain good models for the part where the robot interacts with objects. This results from the fact that the environment is usually not completely known (e.g., position of objects, kinematic structure, physical entities) and that information about how this environment can be manipulated into a certain goal state is not available.

6.1. Evaluation on a synthetic benchmark

In this evaluation, we compare different algorithms on a synthetic benchmark problem. To allow for reproducible quantitative comparison, we define a generalized reinforcement learning problem in the form of equation (18) with parameters $\bar{\boldsymbol{x}} \in \mathbb{R}^2$ and projection $\boldsymbol{\theta} \in \mathbb{R}$. The problem is defined with an analytic cost

$$J(\bar{\boldsymbol{x}}) = (\bar{\boldsymbol{x}}_1^2 + \bar{\boldsymbol{x}}_2^2 - 1)^2 \tag{29}$$

a black-box return

$$R(\boldsymbol{\theta}) = -0.5\boldsymbol{\theta}^2 + \cos(3\boldsymbol{\theta}) \tag{30}$$

and a black-box success

$$S(\boldsymbol{\theta}) = [-1.5 < \boldsymbol{\theta} < 2.5] \tag{31}$$

The projection is defined with the constraint

$$h(\bar{\boldsymbol{x}}, \boldsymbol{\theta}) = \boldsymbol{\theta} - \operatorname{atan}\left(\frac{\bar{\boldsymbol{x}}_1}{\bar{\boldsymbol{x}}_2}\right) \tag{32}$$

The total objective we want to minimize is $J(\bar{\boldsymbol{x}}) - R(\boldsymbol{\theta})$ under the constraint that $S(\boldsymbol{\theta}) = 1$ (see equation (18)). We limit the search space to the region $\bar{\boldsymbol{x}} \in [-1.5, 1.5] \times [-1.5, 1.5]$. This problem has multiple local optima and a global optimum at $\bar{\boldsymbol{x}}^\star = (1, 0)$ with a value of -1. The contours of $J(\bar{\boldsymbol{x}}) - R(\boldsymbol{\theta})$ are visualized in Figure 5, where the red area denotes the infeasible region and the green cross the optima. We compare two different types of algorithm with each other. The first type uses the CORL framework proposed in this paper with different reinforcement learning algorithms (noted as CORL + <method>). The second type are standard reinforcement learning methods that optimize $\bar{\boldsymbol{x}}$ and ignore the specific problem structure. Here is a summary of all evaluated algorithm configurations:

- *PIBU.* Bayesian optimization with the acquisition function PIBU (see equation (24));
- *PoWER.* Policy learning by weighting exploration with the returns (Kober and Peters, 2008);
- *UCB.* Bayesian optimization with the acquisition function upper confidence bound (Brochu et al., 2010);
- *CMA-ES.* Covariance matrix adaptation evolution strategy (Hansen and Ostermeier, 2001);
- *CORL + PIBU.* CORL algorithm with PIBU;
- *CORL + UCB.* CORL algorithm with UCB;
- *CORL + CMA-ES.* CORL algorithm with CMA-ES;
- *CORL + SAL.* CORL algorithm with safe active learning (Schreiter et al., 2015).

Only the PIBU and SAL variants aim for safe exploration during the optimization process. Note that SAL assumes that the distance to the feasibility boundary is observed in critical (but feasible) regions, which all other methods do not make this observation.

To enable those methods to be tested that cannot cope with different objectives and success constraints (i.e., CMA-ES, UCB, and PoWER), we defined the combined objective function

$$o(\bar{\boldsymbol{x}}) = [S(\boldsymbol{\theta}) = 1](J(\bar{\boldsymbol{x}}) - R(\boldsymbol{\theta})) + [S(\boldsymbol{\theta}) = 0]15 \tag{33}$$

The return and cost function can only be observed for parameters that lead to success, such that it is consistent with our method. Failures receive a constant cost of 15. The optimization step in CORL is made with a Newton method. The acquisition function used for the regression Gaussian process g_R uses the hyperparameter $l = 0.4$, $\sigma_{\text{sf}} = 10$ and $\sigma = 0.11$. The classification Gaussian process g_S uses $l = 0.4$, $\sigma_{\text{sf}} = 10$, and a constant prior mean of -7. We

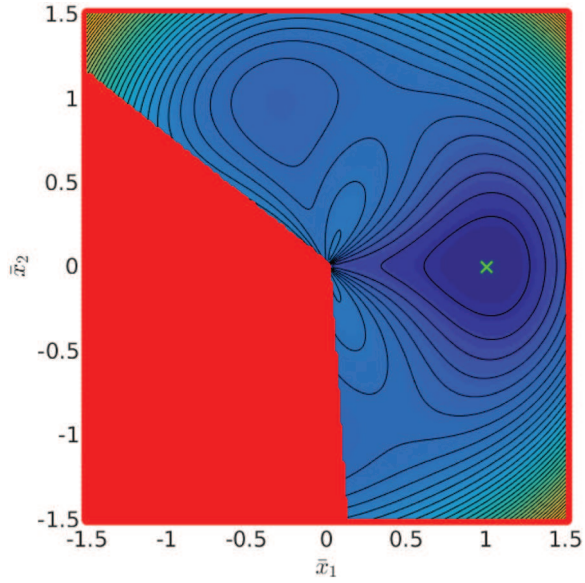


Fig. 5. Contours of $J(\bar{\boldsymbol{x}}) - R(\boldsymbol{\theta})$.

executed all algorithms on this problem from 100 different initial parameters $\bar{\boldsymbol{x}}$, which are samples uniformly in the feasible search region. The CMA-ES algorithm uses a population size of six and the number of offspring is three. Table 1 shows the results of this experiment. We compare the following metrics:

- *Global optimum found.* This metric describes how many times the algorithm found the global optimum $\bar{\boldsymbol{x}}^\star$.
- *Maximum distance to safe region.* The maximum distance of all failure samples to the safety region. All values are given by mean and standard deviation over the 100 trials.
- *Number of failures.* The number of failure samples $S(\boldsymbol{\theta}) = 0$ that were selected by the algorithm until convergence. All values are given by the mean and standard deviation over the 100 trials.

The best two algorithms of each metric are marked in bold in Table 1. The CORL + SAL method is, as expected, the safest method with a mean of 1.57 failure samples—but recall that it assumes it can observe the distance to the boundary in critical regions, which ours does not. Our proposed methods CORL + PIBU find the global optimum very often and exhibit a very low number of near-boundary failures even without observing critical distance. The methods that do not take safety into account give a higher number of failure samples (between five and ten) that are also located far away from the safety region.

6.2. Cabinet

We apply Algorithm 1 in this experiment for the manipulation skill of opening a cabinet using a PR2. The experimental setup is shown in Figure 2. The skill consists of grasping the doorknob, rotating the doorknob until it

Table 1. Results of synthetic benchmark experiment (see Section 6.1).

Method	Global optimum found	Maximum distance to safe region	Number of failures
PIBU	99/100	0.64 ± 0.40	5.27 ± 0.68
PoWER	88/100	1.12 ± 0.44	6.95 ± 4.45
UCB	95/100	1.48 ± 0.11	14.53 ± 1.08
CMA-ES	85/100	1.20 ± 0.38	7.19 ± 4.50
CORL + PIBU	**100/100**	$\mathbf{0.10 \pm 0.04}$	2.05 ± 0.26
CORL + UCB	**100 / 100**	1.26 ± 0.69	$\mathbf{1.38 \pm 0.98}$
CORL + CMA-ES	95/100	0.97 ± 0.53	3.73 ± 2.53
CORL + SAL	96/100	$\mathbf{0.06 \pm 0.12}$	$\mathbf{1.57 \pm 3.38}$

unlocks the door joint and opening the door. The full policy parametrization is a trajectory $\bar{\boldsymbol{x}}$ that consists of 200 time steps and 11 degrees of freedom (nine belong to the robot and two to the cabinet). The trajectory is executed with a duration of 15 s. We recorded a single demonstration with kinesthetic teaching as initialization.

We define the low-dimensional $\boldsymbol{\theta}$ as described in Section 5.1. We select a three-dimensional space of the interaction with the cabinet. The first parameter is the opening angle of the cabinet at the end of the manipulation. The second parameter is the reference gripper opening, which corresponds to the amount of force that is used while grasping the doorknob. The third parameter is the final angle of the doorknob, which corresponds to whether the cabinet door can be manipulated. All parameters are defined relative to the initial demonstration. The analytic cost function J measures the sum of squared accelerations over the complete trajectory. The black-box return R is the negative amount of force used during the opening, which is measured using a force or torque sensor in the wrist of the robot. The black-box success S is the binary signal if the door was opened successful to a certain degree.

In the first part of Algorithm 1, we use CORL to improve the skill with respect to our defined objectives. We compare four different algorithm configurations of CORL for this problem:

1. *CORL + PIBU ($l = 0.3$).* Bayesian optimization with the acquisition function PIBU and a wide kernel length scale l. The hyperparameters are $\sigma_{\text{sf}} = 0.5$ and $\sigma = 0.01$ for the regression Gaussian process g_R and $\sigma_{\text{sf}} = 6$ for the classification Gaussian process g_S.
2. *CORL + PIBU ($l = 0.15$).* PIBU with a narrow kernel length scale l. The other hyperparameters are identical to configuration 1.
3. *CORL + CMA-ES ($\sigma = 0.3$).* CMA-ES with a high initial variance σ^2, a population size of seven and three parents.
4. *CORL + CMA-ES ($\sigma = 0.1$).* CMA-ES with a small initial variance σ^2. The other parameters are identical to configuration 3.

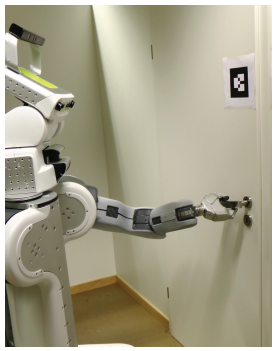
In CORL + PIBU, we use two different hyperparameters l for the kernel in equation (23), which corresponds to how far a datapoint extrapolates its value. In the CMA-ES case, we use a configuration with a small initial variance and a configuration with a wide initial variance of the samples. The results of the experiment are given in Figure 3. Figure 3(a) shows the three-dimensional projection space $\boldsymbol{\theta}$ for the configuration CORL + PIBU ($l = 0.3$). The success region $g_S(\boldsymbol{\theta}) = 0$ is visualized as a gray surface; all the points inside lead to success and the points outside to failure. The successful samples are visualized as dots; the color denotes the return value. Blue dots have the lowest and red dots the highest return value. Failures are visualized as crosses. The best candidate is visualized with a green circle around it.

Figure 3(b) shows the best candidate and Figure 3(c) shows the success rate over iterations. All methods lead to a similar best policy; the learning behavior depends on the hyperparameter of each method. The best policy receives an objective of 4.33, where the analytic cost $J(\bar{\boldsymbol{x}})$ is 0.96 and the black-box return $R(\boldsymbol{\theta})$ is -3.37. The configuration CORL + PIBU ($l = 0.30$) already finds its best solution after 20 iterations, but requires more iterations until convergence to explore the whole success region. The configuration CORL + PIBU ($l = 0.15$) leads to the highest success rate of 0.8, but also requires more iterations until convergence than other variants. This result shows the tradeoff that must be made between convergence speed and safe exploration.

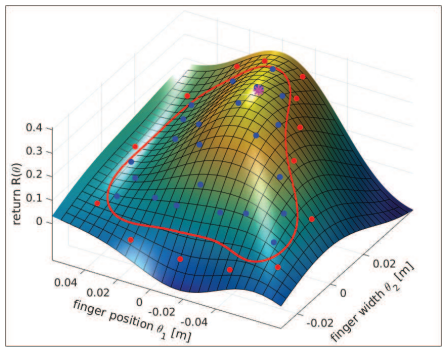
In the second part of Algorithm 1, we use the collected data to learn a higher-level cost-function representation of the skill. We use three successful trajectories from the dataset D and apply the IKKT algorithm with the features described in Section 5.2. The resulting cost function is able to generalize to different initial positions and door angles of the cabinet (see Figure 3(d) and (e)).

6.3. Door

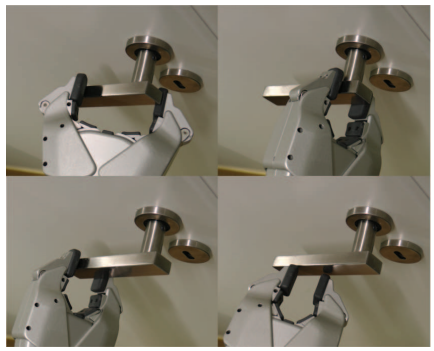
In this experiment, we apply Algorithm 1 on opening a door (see Figure 6(a)). The motion also includes the unlocking of the door by turning the handle first. We define two parameters in the contact space of the door handle as low-dimensional parameter $\boldsymbol{\theta}$. The first parameter is the finger position on the handle relative to the demonstration. The second parameter is the finger opening width. Different grasps of such a projection are shown in Figure 6(c).



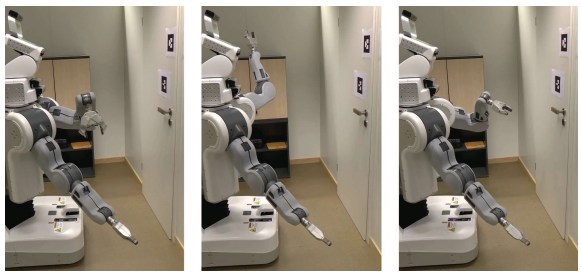
(a) Setup.



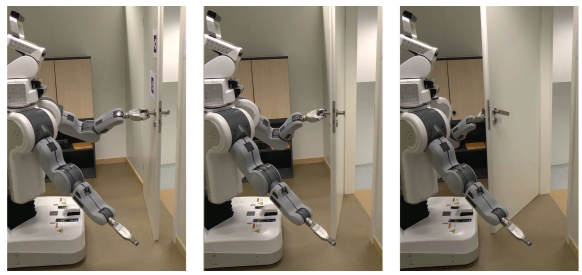
(b) Return function.



(c) Projection.



(d) Different initial positions.



(e) Different final door angles.

Fig. 6. Door experiment (see Section 6.3). (a) Setup of the PR2 opening a door. (b) Learned return function $R(\boldsymbol{\theta})$ with Bayesian optimization. Blue points denote successful rollouts, red points denote failures, and the magenta star is the best parameter found. The red line denotes the decision boundary of the classifier. (c) Four different grasps that were tried during learning. (d, e) Generalization abilities of our approach regarding different initial positions of the robot and different final door angles. After learning the weight parameter $\boldsymbol{w}^\star$ with IKKT, it was possible to generalize to all these scenarios for the door opening.

We use the same objectives $J(\bar{\boldsymbol{x}})$, $R(\boldsymbol{\theta})$, and $S(\boldsymbol{\theta})$ as in the previous experiment. To achieve autonomous learning, we used markers on the door to measure if it was opened successfully and added a simple motion that closed the door after each trial. This allowed the robot to perform the learning on its own without human intervention. The parameters of the regression Gaussian process g_R that we used were $l = 0.042$, $\sigma_{\text{sf}} = 0.168$, and $\sigma = 0.012$. We set $l = 0.02$, $\sigma_{\text{sf}} = 10$, and a constant prior mean of -7 for the classification Gaussian process g_S.

We compared our method with a CORL + CMA-ES variant, where both CMA-ES and PIBU operate on the low-dimensional projection $\boldsymbol{\theta}$, exploiting the combination with the analytic motion optimization. The resulting return and success function are shown in Figure 6(b). Our method converged in this run after 40 rollouts. From these 40 rollouts, 26 were successes and 14 were failures. The blue dots indicate successful rollouts; the red dots, failures; and the magenta star shows the best parameter. The red line denotes the classifier boundary.

The results are shown in Table 2. We use as a performance measure the highest objective, the failure rate with the system and the maximum distance to the safety region. All values are reported as mean and standard deviation over four runs. It can be seen that CORL + PIBU achieves a lower failure rate with a very low standard deviation. The failures of PIBU are also closer to the safety region than CMA-ES. This results from the fact that the boundary is explored with our acquisition function (see equation (24)). The CORL + CMA-ES method also finds a slightly worse policy than CORL. We tried alternative approaches that do not rely on this low-dimensional projection and the combination with an analytic motion optimizer: We performed experiments with dynamic movement primitives and PoWER similar to those of Kober and Peters (2008). For this we parametrized the shape and goal of the dynamic movement primitive, leading to a 96-dimensional parameter space. However, we could not achieve a noticeable learning performance after 150 iterations. We assume that the black-box return function that combines the amount of forces with path smoothness is not sufficiently informative for this large parameter space. This reinforces the motivation for our general approach of dissecting objectives into high-dimensional analytical and low-dimensional black-box parts.

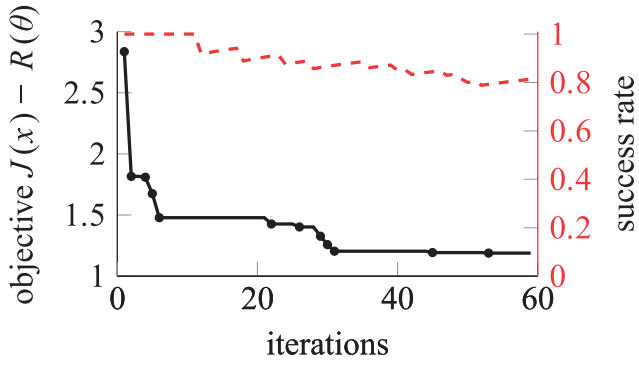
In a previous experiment (Englert and Toussaint, 2015), we applied IKKT on the same action from repeated demonstrations. We were able to generate motions robustly with these parameters, generalized to different initial positions and different final door angles (see Figure 6(d) and (e)). The

Table 2. Evaluation of PIBU / CMA-ES.

Method	Highest return	Failure rate	Maximum distance to safe region
CORL + PIBU	0.45 ± 0.032	0.350 ± 0.025	0.0146 ± 0.006
CORL + CMA-ES	0.41 ± 0.017	0.397 ± 0.131	0.0358 ± 0.030

(a) Sliding skill.

(b) Success constraint motion.



(c) Objective function and success rate.

Fig. 7. Lockbox experiments (see Section 6.4). (a) Sliding skill that we want to learn. (b) Success constraint motion that checks whether the sliding was successful, which means that the next joint can be manipulated. (c) Best objective (black solid line) and success rate (red dashed line) over iterations.

demonstration, learning behavior, and resulting motions are shown in Extension 1.[1]

6.4. Lockbox

A further experiment that we conducted is the manipulation of a linear joint of a lockbox (see Figure 7(a)). The lockbox was designed to research different physical exploration strategies (Baum et al., 2017) and consists of multiple rotational and translational degrees of freedom that lock each other. In this experiment, we focus on a translational joint that we want to open with a sliding motion. The low-dimensional projection is the vertical location of the contact point and the sliding velocity. The goal is to manipulate the translational joint, such that the next joint is unlocked. We evaluate the success constraint by executing another motion that checks whether the next joint can be manipulated (see Figure 7(b)). Further, we also added a motion that closes both joints again, which allowed the robot to achieve complete autonomous learning without human supervision (see Extension 1). We used the same Gaussian process hyper-parameter and objectives as in the previous section. The CORL + PIBU algorithm converged after 59 iterations with 48 successes and 11 failures. The total interaction time of the robot was 61 min. Figure 7(c) shows the learning curve (black line) and success rate (red dashed line).

7. Conclusions

In this work, we presented an approach to learning manipulation skills from a single demonstration with the goal of achieving wide generalization abilities and a good performance. We incorporated the structure of manipulation

skills in our approach by using a low-dimensional projection of the motion that parametrizes the interaction with the environment. We used analytic motion optimization to improve the full motion and episodic reinforcement learning to improve the interactions. The advantage of this separation is that it was not necessary to specify models of the interactions, which is very difficult in practice. Our approach requires as input a single demonstration of the skill that is bootstrapped with reinforcement to become more robust and efficient before inverse optimal control is used to learn a constrained optimization problem. This design allowed us to reduce the required input and human supervision. Using a constrained motion optimization representation allowed us to generalize the skill with respect to different initial states and to control the skill with respect to desired states of the external degrees of freedom.

We evaluated our algorithm using a synthetic benchmark function where we compared it with alternative learning methods. The results indicate that the combination of optimization and learning leads to a faster convergence. The results also denote that the integration of a success constraint results in safer learning by avoiding very bad samples outside the feasible region. We also demonstrated our algorithm using several real-robot experiments. Thereby, we used a variety of different low-dimensional projections to parametrize the interaction with the environment. Our algorithm improved all three skills with respect to the objectives and learned a constrained optimization problem that generalized the skill to new scenarios.

In future work, it is important to investigate generalization abilities regarding the skill transfer to different environments (e.g., different geometrical or physical entities) that avoids learning from scratch for each environment. An interesting question is which kind of representation is most suitable for skill transfer between different environments. Such a transfer would further improve the abilities of robots in our world and focus the learning on fine-tuning for the new environments. Another goal of future research should be toward a more integrated learning algorithm that makes efficient use of the collected data to further improve the skill and knowledge of the world. For example, the data we collect during learning can be used to update the limits, unlocking mechanisms, and location of external degrees of freedom (Kulick et al., 2015; Sturm et al., 2011). A further goal for us is to use the learned skills in higher-level symbolic planning methods. We think that the constrained optimization problem is a suitable representation, since it allows use of the skill inside a symbolic planning problem for a wide range of scenarios.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the DFG (grant number TO 409/9-1).

Note

1. https://youtu.be/sG01B_GcTJQ

References

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the twenty-first international conference on machine learning*, Banff, Canada, 4–8 July 2004. New York: ACM.

Abbeel P, Coates A, and Ng AY (2010) Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29(13): 1608–1639.

Abbeel P, Coates A, Quigley M, et al. (2007) An application of reinforcement learning to aerobatic helicopter flight. In: Schölkopf B, Platt J, and Hofmann T (eds.) *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA: MIT Press, pp. 1–9.

Achiam J, Held D, Tamar A, et al. (2017) Constrained policy optimization. In: *34th international conference on machine learning*, Sydney Australia, 6–11 August 2017. Piscataway, NJ: IEEE.

Argall BD, Chernova S, Veloso M, et al. (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.

Baum M, Bernstein M, Martín-Martín R, et al. (2017) Opening a lockbox through physical exploration. In: *Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids 2017)*, Birmingham, UK, 15–17 November. Piscataway, NJ: IEEE.

Berkenkamp F and Schoellig AP (2015) Safe and robust learning control with Gaussian processes. In: *Proceedings of European control conference*, Linz, Austria, 15–17 July 2015. Piscataway, NJ: IEEE.

Bristow DA, Tharayil M, and Alleyne AG (2006) A survey of iterative learning control. *IEEE Control Systems* 26(3): 96–114.

Brochu E, Cora VM, and De Freitas N (2010) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv* 1012.2599 [cs.LG].

Calandra R, Seyfarth A, Peters J, et al. (2015) Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence* 76(1): 5–23.

Chebotar Y, Kalakrishnan M, Yahya A, et al. (2017) Path integral guided policy search. In: *Proceedings of the international conference on robotics and automation*, Singapore, 29 May–3 June 2017. Piscataway, NJ: IEEE.

Englert P and Toussaint M (2015) Inverse KKT—learning cost functions of manipulation tasks from demonstrations. In: Bicchi A and Burgard W (eds.) *Robotics Research*. (*Springer Proceedings in Advanced Robotics*, vol 3). Cham: Springer, pp. 57–72.

Englert P and Toussaint M (2016) Combined optimization and reinforcement learning for manipulations skills. In: *Robotics: science and systems* (ed. D Hsu, N Amato, S Berman, et al.), Ann Arbor, MI, 18–22 June 2016. Available at: http://www.roboticsproceedings.org/rss12/index.html

Finn C, Levine S, and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: *International conference on machine learning* (eds. MF Balcan and KQ Weinberger), New York, NY, 29–14 June 2016, pp. 49–58. Brookline, MA: Microtome Publishing.

Fu J, Levine S, and Abbeel P (2016) One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In: *IEEE/RSJ international conference on intelligent robots and systems*, Daejeon, South Korea, 9–14 October 2016. Piscataway, NJ: IEEE.

Garcia Polo FJ and Fernandez-Rebollo F (2011) Safe reinforcement learning in high-risk tasks through policy improvement. In: *IEEE symposium on adaptive dynamic programming and reinforcement learning*. Paris, France, 11–15 April 2011, Piscataway, NJ: IEEE.

Gardner J, Kusner M, Xu Z, et al. (2014) Bayesian optimization with inequality constraints. *Proceedings of Machine Learning Research* 32(2): 937–945.

Gelbart MA, Snoek J, and Adams RP (2014) Bayesian optimization with unknown constraints. In: *UAI'14 proceedings of the thirtieth conference on uncertainty in artificial intelligence* (ed. N Zhang and J Tian), Quebec City, Canada, 23–27 July 2014, pp. 250–259. Arlington, VA: AUAI Press.

Gramacy RB and Lee HKH (2011) Optimization under unknown constraints. In: Bernardo JM, Bayarri MJ, and Berger JO (eds.) *Bayesian Statistics 9: Proceedings of the Ninth Valencia International Meeting*. Oxford, UK: Oxford University Press, pp. 229–256.

Hansen N and Ostermeier A (2001) Completely derandomized self-adaptation in evolution. Strategies. *Evolutionary Computation* 9(2): 159–195.

Kalakrishnan M, Pastor P, Righetti L, et al. (2013) Learning objective functions for manipulation. In: *IEEE international conference on robotics and automation*, Karlsruhe, Germany, 6–10 May 2013. Piscataway, NJ: IEEE.

Kalakrishnan M, Righetti L, Pastor P, et al. (2011) Learning force control policies for compliant manipulation. In: *International conference on intelligent robots and systems*, San Francisco, CA, 25–30 September 2011. Piscataway, NJ: IEEE.

Kober J and Peters J (2008) Policy search for motor primitives in robotics. In: *NIPS'08 proceedings of the 21st international conference on neural information processing systems* (ed. D Koller, D Schuurmans, Y Bengio et al.), Vancouver, Canada, 8–10 December 2008, pp. 849–856. Red Hook, NY: Curran Associates, Inc.

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

Kolter JZ, Abbeel P, and Ng AY (2008) Hierarchical apprenticeship learning with application to quadruped locomotion. In: *NIPS'07 Proceedings of the 20th international conference on neural information processing systems* (ed. JC Platt, D Koller, Y Singer, et al.), Vancouver, Canada, 3–6 December 2007, pp. 769–776. Red Hook, NY: Curran Associates, Inc.

Kulick J, Otte S, and Toussaint M (2015) Active exploration of joint dependency structures. In: *International conference on robotics and automation*, Seattle, WA, 26–30 May 2015. Piscataway, NJ: IEEE.

Kupcsik AG, Deisenroth MP, Peters J, et al. (2013) Data-efficient generalization of robot skills with contextual policy search. In: *AAAI'13 proceedings of the twenty-seventh AAAI conference on artificial intelligence*, Bellevue, WA, 14–18 July 2013, pp. 1401–1407. Palo Alto, CA: AAAI Press.

Kushner HJ (1964) A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering* 86(1): 97–106.

Levine S and Koltun V (2012) Continuous inverse optimal control with locally optimal examples. In: *ICML'12 proceedings of the 29th international conference on machine learning*, Edinburgh, UK, 26 June–1 July 2012, pp. 475–482. Madison, WI: Omnipress.

Levine S and Koltun V (2013) Guided policy search. *Proceedings of Machine Learning Research* 28(3): 1–9

Levine S, Finn C, Darrell T, et al. (2016) End-to-end training of deep visuomotor policies *The Journal of Machine Learning Research* 17(1): 1334-13730.

Levine S, Popovic Z, and Koltun V (2011) Nonlinear inverse reinforcement learning with Gaussian processes. In: *NIPS'11 proceedings of the 24th international conference on neural information processing systems* (eds. J Shawe-Taylor, RS Zemel, PL Bartlett, et al.), Granada, Spain, 12–15 December 2011, pp. 19–27. Red Hook, NY: Curran Associates, Inc.

Lizotte DJ, Wang T, Bowling MH, et al. (2007) Automatic gait optimization with Gaussian process regression. In: *International joint conference on artificial intelligence* (ed. R Sangal, H Mehta, and RK Bagga), Hyderabad, India, 6–12 January 2007, pp. 944–949. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Mockus J, Tiešis V, and Žilinskas A (1978) The application of Bayesian methods for seeking the extremum. In: Dixon LCW and Szego GP (eds.) *Towards Global Optimization*, vol. 2. Amsterdam: North-Holland, pp. 117–129.

Muelling K, Kober J, Kroemer O, et al. (2013) Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research* (3): 263–279.

Ng AY and Russell S (2000) Algorithms for inverse reinforcement learning. In: *ICML'00 proceedings of the seventeenth international conference on machine learning* (ed. P Langley), Stanford, CA 29 June–2 July 2000. pp. 663–670. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Nickisch H and Rasmussen CE (2008) Approximations for binary Gaussian process classification. *Journal of Machine Learning Research* 9(10): 2035–2078.

Peters J, Mülling K, and Altun Y (2010) Relative entropy policy search. In: *AAAI'10 proceedings of the twenty-fourth AAAI conference on artificial intelligence*, Atlanta, GA, 11–15 July 2010. Palo Alto, CA: AAAI Press.

Puydupin-Jamin AS, Johnson M, and Bretl T (2012) A convex approach to inverse optimal control and its application to modeling human locomotion. In: *IEEE international conference on robotics and automation*, Saint Paul, MN, 14-18 May 2012. Piscataway, NJ: IEEE.

Rasmussen CE and Williams CKI (2006) *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Rückert EA, Neumann G, Toussaint M, et al. (2013) Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience* 6(138): 97.

Schonlau M, Welch WJ, and Jones DR (1998) Global versus local search in constrained optimization of computer models. *Lecture Notes—Monograph Series* 34: 11–25.

Schreiter J, Nguyen-Tuong D, Eberts M, et al. (2015) Safe exploration for active learning with Gaussian processes. In: Bifet A, May M, Zadrozny B, et al. (eds.) *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2015* (*Lecture Notes in Computer Science*, vol. 9286). Cham: Springer, pp. 133–149.

Stulp F and Sigaud O (2013a) Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics* 4(1): 49–61.

Stulp F and Sigaud O (2013b) Policy improvement methods: Between black-box optimization and episodic reinforcement learning. *Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes*, Lille, France, 1–2 July 2013.

Sturm J, Stachniss C, and Burgard W (2011) A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research* 41: 477–526.

Sui Y, Gotovos A, Burdick JW, et al. (2015) Safe exploration for optimization with Gaussian processes. In: *ICML'15 proceedings of the 32nd international conference on machine learning*, Lille, France, 6–11 2015, vol. 37, pp. 997–1005. Brookline, MA: Microtome Publishing.

Sutton RS and Barto AG (1998) *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Theodorou E, Buchli J, and Schaal S (2010) A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research* 11: 3137–3181.

Toussaint M (2017) A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian process smoothing, optimal control, and probabilistic inference. In: Laumond J-P, Mansard N, and Lasserre J-B (eds.) *Geometric and Numerical Foundations of Movements*. Cham: Springer, pp. 361–392.

Toussaint M, Ratliff N, Bohg J, et al. (2014) Dual execution of optimized contact interaction trajectories. In: *Proceedings of the international conference on robotics and automation*, Chicago, IL, 14–18 September 2014. Piscataway, NJ: IEEE.

Vuga R, Nemec B, and Ude A (2015) Enhanced policy adaptation through directed explorative learning. *International Journal of Humanoid Robotics* 12(3): 1550028.

Wright SJ and Nocedal J (1999) *Numerical Optimization*, vol. 2. New York: Springer.

Zhifei S and Joo EM (2012) A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics* 5(3): 293–311.

Ziebart BD, Maas A, Bagnell JA, et al. AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI'08 proceedings of the 23rd AAAI national conference on artificial intelligence*, Chicago, IL, 13–17 July 2008 vol. 3, pp. 1433–1438. Palo Alto, CA: AAAI Press.

Appendix A: Index to multimedia extension

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of multimedia extension.

Extension	Media type	Description
1	Video	Demonstration, learning behavior, and resulting motions