Combined Optimization and Reinforcement Learning for Manipulation Skills

Peter Englert and Marc Toussaint
Machine Learning & Robotics Lab
University of Stuttgart, Germany

Abstract—**This work addresses the problem of how a robot can improve a manipulation skill in a sample-efficient and secure manner. As an alternative to the standard reinforcement learning formulation where all objectives are defined in a single reward function, we propose a generalized formulation that consists of three components: 1) A *known* analytic control cost function; 2) A black-box return function; and 3) A black-box binary success constraint. While the overall policy optimization problem is high-dimensional, in typical robot manipulation problems we can assume that the black-box return and constraint only depend on a lower-dimensional projection of the solution. With our formulation we can exploit this structure for a sample-efficient learning framework that iteratively improves the policy with respect to the objective functions under the success constraint. We employ efficient 2nd-order optimization methods to optimize the high-dimensional policy w.r.t. the analytic cost function while keeping the lower dimensional projection fixed. This is alternated with safe Bayesian optimization over the lower-dimensional projection to address the black-box return and success constraint. During both improvement steps the success constraint is used to keep the optimization in a secure region and to clearly distinguish between motions that lead to success or failure. The learning algorithm is evaluated on a simulated benchmark problem and a door opening task with a PR2.**

I. Introduction

In this paper we consider the problem of how a robot can improve its performance on a manipulation skill. Every real-world interaction with the environment only gives a single data point w.r.t. the success or failure of the manipulation. To improve a high-dimensional skill policy from so little data we need to exploit prior assumptions about the structure of the problem. One interesting assumption is that for some aspects of the problem we actually have models and analytic cost functions, while other aspects, like the overall success or reward, are black-boxes without known models.

We would like to exemplify this with the task of opening a door, which consist of different subgoals such as reaching the handle, pushing the handle down and performing the opening motion while keeping contact. After a manipulation trial the robot receives a single Boolean success signal on whether the door opened. Further objectives are a smooth motion and collision avoidance. Combining these objectives into a single reward function and optimizing it, e.g. with policy search methods, discards the structure of the problem and is sample-inefficient for high-dimensional policies. Generally, in typical manipulation problems, a kinematic or dynamic model of the robot itself is available, wherease the environment (e.g., contact interactions, door joints) may be unknown or difficult to model accurately, forbidding an analytic model for the overall manipulation success. In our example the smoothness term can be expressed analytically, whereas the feedback if the door is open or not is only available as a Boolean black-box function. It is the aim of our framework to exploit such structural knowledge for the sake of sample efficiency.

In this paper we propose a generalized formulation of reinforcement learning that allows to express more structure in the problem definition. In this formulation we assume the objectives are defined with three components: An analytic cost function, a black-box return function and a black-box binary success constraint. The goal of this structure is to separate the objective terms in such a way that it is possible to exploit knowledge (e.g., analytic form, models) when it is available as well as to explore in a model-free way otherwise (i.e., black-box objectives). For black-box functions it is difficult to optimize them efficiently in high-dimensional parameter spaces. Therefore, we make the second assumption that the black-box return and success constraint only depend on a lower dimensional projection of the policy. This lower dimensional projection additionally encodes crucial structure of the problem. We make the assumption that it is given and discuss which projection we typically have in mind in the context of manipulation. For instance, the success of pressing the door handle essentially depends on the placement of the contact points rather than the full reaching motion—such interaction parameters typically define the lower dimensional projection we consider.

The starting point for learning is a single demonstration of successful (but non-efficient) manipulation (e.g., given by kinesthetic demonstration or hand-programmed). From this initialization the robot improves the skill with respect to the given objective functions and task success constraint. We alternate between two improvement strategies to achieve this goal. The first improvement strategy is an efficient Gauss-Newton motion optimization that optimizes the analytic cost function of the motion and implies a policy following this reference. Here we fix the lower dimensional projection by incorporating them as hard constraints into the mathematical program. The second improvement strategy is Bayesian optimization over

This work was supported by the EU-ICT project 3rdHand 610878 and the DFG project Exploration Challenge BR 2248/3-1.
Email: peter.englert@informatik.uni-stuttgart.de
marc.toussaint@informatik.uni-stuttgart.de

the lower dimensional projection, which aims at maximizing the black-box return function subject to the unknown success constraint. We use Gaussian processes to learn models for the black-box objective function and constraint. These models are combined in a novel acquisition function to select the next policy in a secure and efficient manner.

The main contributions of this paper are:

1) The more structured reinforcement learning formulation with an analytic cost function, a black-box return function and a black-box success constraint.
2) A policy improvement framework CORL that combines optimization and black-box reinforcement learning.
3) A novel acquisition function PIBU for Bayesian optimization that selects the next policy in a secure and efficient manner.

In the next section we present related work. We introduce in Section III background on optimal control and episodic reinforcement learning. In the Sections IV–VII we present our problem formulation and describe the CORL algorithm. In Section VIII we evaluate our approach on a synthetic problem and real-world task of opening a door with a PR2.

II. Related Work

A. Policy Search in Robotics

Reinforcement learning is a widely used approach for learning skills in robotics [12]. Kober et al. [11] proposed to use dynamic movement primitives as policy representation and the policy search method PoWER to learn the shape and properties of the motion. Usually this approach is initialized from demonstration and afterwards policy search methods improve the skill with respect to a defined reward function. The difference to our approach is that we perform learning on two policy parameterizations. This allows us to use efficient Gauss-Newton optimization routines for the parts of a motion where an analytic cost function is available.

Kalakrishnan et al. [10] present a method to learn force control policies for manipulations. The policy is initialized with position control via imitation and afterwards this policy is augmented with a force profile that is learned with the reinforcement learning method PI2 [26]. They use a single reward function that combines smoothness terms, force terms and tracking errors to the demonstration. The weighting of the different terms in the cost function is non-trivial. Again, this method builds on dynamic movement primitives as representation of non-stationary policies instead of exploiting optimization w.r.t. the known analytic objectives to optimize the non-stationary policy. In our experiments we compare to CMA, which has been shown to be closely related to PI2 [23].

B. Bayesian Optimization with Constraints

There are many approaches on how to incorporate constraints into Bayesian optimization [6, 7, 8, 21]. The core concept of these approaches is to use the probability of a constraint being fulfilled and combine it with other acquisition functions (e.g., expected improvement). There exist multiple variations on how to formulate the acquisition function depending on the problem requirements. In Gardner et al. [6], for example, they assume that the objective can also be evaluated in the unfeasible region and in Gelbart et al. [7] they consider the case of a decoupled observation of objective and constraint.

The main difference to our work is that we additionally want to have a secure learning process which avoids sampling constraint violating points. To obtain this we propose a novel acquisition function that uses the variance of the constraint to guide the exploration on the decision boundary.

C. Safe Exploration

Schreiter et al. [22] propose a safe exploration strategy SAL for a similar problem to ours. They optimize a function in a safe manner where the feasible region is unknown. For doing this they assume to observe a safety measure when samples are close to the boundary. This information is integrated into a differential entropy exploration criteria to select next candidates. They also provide an upper bound for the probability of failure. This approach would most likely lead to fewer failures during the exploration than ours, but it requires the additional observation of a safety measure close to the decision boundary which is not available in our problem formulation. We compare our approach to this strategy on a synthetic problem in Section VIII-A.

Another approach for a safe exploration is proposed in Sui et al. [25]. The strategy SAFEOPT 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. They 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 is used to learn a stabilization task on a quadrotor vehicle in [1].

None of the above methods for safe or Bayesian exploration would be sample-efficient when directly applied on the high-dimensional non-stationary policy. However, they can be integrated in our proposed CORL framework, as we will demonstrate for UCB and SAL in the evaluations.

D. Combined Learning Approaches

There has been previous work on how reinforcement learning methods can be combined with optimization to more powerful algorithms. In Rückert et al. [20] a reinforcement learning algorithm for planning movement primitives is introduced that uses a two-layered learning formulation. In an outer loop the policy search method CMA optimizes an extrinsic cost function that measures the task performance. This policy search is 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 to our approach is that they 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 sequentially. Additionally, we use a safety constraint to guide the learning in a secure manner. Another approach that combines reinforcement learning with optimization was proposed by Vuga et al. [28]. They combine iterative learning control with the reinforcement learning algorithm PI2. Our approach differs especially w.r.t. the lower-dimensional structure and that we consider arbitrary black-box objectives.

Kupcsik et al. [13] propose a policy search method that combines model-free reinforcement learning with learned forward models. They learn probabilistic forward models of the robot and the task 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 is used to maximize the reward and balance the exploration and experience loss by staying close to the observed data. One of the main differences to our approach is that we divide the problem in model-based motion optimization that improves the motion efficiently and reinforcement learning that improves the task by exploring a lower-dimension representation. A further difference is that they learn a model of the task that is used in internal simulations, whereas we directly learn a model that maps parameters to returns.

III. Background on Optimal Control and Episodic Reinforcement Learning

We consider non-stationary policies $\pi(x_t, t, w)$ parametrized by a parameter vector $w \in \mathbb{R}^n$ that map a system state $x_t \in \mathbb{R}^Q$ at time t to an action $u_t \in \mathbb{R}^U$. In finite-horizon stochastic optimal control with discretized time [2], we assume that the transition model $P(x_{t+1} | x_t, u_t)$ is known, as well as the objective function

$$J(w) = \mathbb{E}\left[\sum_{t=0}^{T} c_t(x_t, u_t) \,|\, \pi\right] . \tag{1}$$

In the case of linear dynamics, quadratic costs and Gaussian noise (LQG) the problem can be solved analytically. In the non-linear case one approach is a Laplace approximation, which first computes the most likely path leveraging classical motion optimization (e.g., KOMO [27]) and then approximates and solves the LQG problem around the path (equivalent to iLQG).

When the system model and objective function are unknown and estimating the model or a value function does not seem promising, policy search for episodic reinforcement learning (RL) aims at finding optimal policy parameters w for unknown stationary rewards $r(x,u)$ by maximizing the return $R(w) = \mathbb{E}[\sum_{t=0}^{T} r(x_t, u_t) | \pi]$. In contrast to optimal control, the system behavior as well as rewards can only be observed by doing rollouts on the real system. Standard methods for such policy search are PoWER [11] and PI2 [26].

A restricted case of episodic reinforcement learning is where only the total return $\sum_{t=0}^{T} r(x_t, u_t)$ of an episode is observed but not the individual rewards $r(x_t, u_t)$ [24]. In this case black-box optimization methods can be used to solve the problem (e.g., covariance matrix adaptation [9] or Bayesian optimization [15]). Our approach combines this type of episodic reinforcement learning with non-linear optimal control based on motion optimization.

IV. Combined Optimization and Reinforcement Learning

We propose a structured RL formulation that combines optimal control and episodic RL. We specifically aim to deal with cases where the policy parameters w 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.

Our problem formulation consists of (i) an *analytically known cost function* $J(w)$ (cf. Equation (1)), (ii) a q-dimensional *projection constraint*

$$h(w, \theta) = 0 \ , \quad h : \ \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q \tag{2}$$

that ties every policy parameter w to a lower-dimensional projection $\theta \in \mathbb{R}^m$ (see details below), (iii) an *unknown black-box return function*

$$R(\theta) : \mathbb{R}^m \to \mathbb{R} \ , \tag{3}$$

and (iv) an *unknown black-box success constraint*

$$S(\theta) : \mathbb{R}^m \to \{0, 1\} \ . \tag{4}$$

The generalized reinforcement learning problem is

$$\min_{w,\theta} \ J(w) - R(\theta) \quad \text{s.t.} \quad h(w,\theta) = 0 \ , \quad S(\theta) = 1 \ . \tag{5}$$

That is, we want the best policy parameters $(w^\star, \theta^\star)$ (measured with $J(w^\star)$ and $R(\theta^\star)$) that fulfill a task (measured with $S(\theta^\star) = 1$).

The projection constraint $h(w, \theta)$ defines the relation between the high-dimensional w and the lower dimensional θ. We generally assume that h is smooth and that, for given w, $h(w, \theta) = 0$ identifies a *unique* $\Theta(w) = \theta$. In that sense, θ is a projection of w. In general, θ should represent those aspects of the policy that are crucial for success but for which analytical objective or success functions are not available. In the context of manipulation, interaction parameters such as the contact points of a grasp exemplify such aspects. In Section V-B we will explain specifically how we choose $h(w, \theta)$ for robot manipulation tasks.

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

We solve the problem in Equation (5) by using optimal control methods to improve the policy w.r.t. the high-dimensional w and black-box Bayesian optimization to improve the policy w.r.t. the low-dimensional θ. The resulting algorithm CORL is summarized in Algorithm 1.

As input to our method we assume to have an initial policy parameterization $(w^{(0)}, \theta^{(0)})$ that fulfills the task ($S(\theta^{(0)}) = 1$).

Algorithm 1: CORL
Combined Optimization and Reinforcment Learning

```
input: Initial parametrization $(w^{(0)}, \theta^{(0)})$
       Analytically known: $J, h$
       As black-box functions: $R, S$
Initialize $(w^\star, \theta^\star) = (w^{(0)}, \theta^{(0)})$, $i = 0$
repeat
    1) Optimization (Section VI)
    repeat
        $w^{(i+1)} = \operatorname{argmin}_w J(w)$
        s.t.  $h(w, \theta^\star) = 0, \quad ||w - w^{(i)}|| < \varepsilon_{\max}$
        Perform rollout with policy parameter $w^{(i+1)}$
        Set best $w^\star = w^{(i+1)}$ if rollout is successful
    until $||w^{(i+1)} - w^{(i)}|| < \varepsilon_w$
    2) Black-box Reinforcement Learning (Section VII)
    Initialize $D = \{\theta^\star, R(\theta^\star), S(\theta^\star)\}$, $d = 0$
    repeat
        $\theta^{(d+1)} = \operatorname{argmax}_\theta a(\theta, D)$
        $w^{(i+1)} = \operatorname{argmin}_w ||w^\star - w||^2$ s.t. $h(w, \theta^{(d+1)}) = 0$
        Perform rollout with policy parameter $w^{(i+1)}$
        Add datapoint $\{\theta^{(d+1)}, R(\theta^{(d+1)}), S(\theta^{(d+1)})\}$ to $D$
    until $||\theta^{(d+1)} - \theta^{(d)}|| < \varepsilon_\theta$
    Set best $\theta^\star = \operatorname{argmax}_{\theta \in D} R(\theta)$  s.t.  $S(\theta^\star) = 1$
until no change in policy parameter $(w^\star, \theta^\star)$
```

The first improvement strategy is constrained optimization and acts on the high-dimensional w to improve the analytic cost function $J(w)$. Thereby the lower dimensional parameter θ is kept fixed with the equality constraint $h(w, \theta) = 0$. Fixing the lower dimensional parameter θ means that the resulting policy fulfills a certain property that is defined by $h(w, \theta)$. We assume that the task success only depends on θ, which implies that all policy parameters w that fulfill the constraint for a fixed θ lead to the same success outcome.

The second improvement strategy in Algorithm 1 is black-box reinforcement learning over θ that aims at improving $R(\theta)$ and fulfilling the constraint $S(\theta)$. We propose a new acquisition function $a(\theta)$ for Bayesian optimization that is optimized in each iteration to select the next parameter. We define $a(\theta)$ 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 $S(\theta)$ 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.

At the end of each improvement step in Algorithm 1, we select the best policy parameter $(w^\star, \theta^\star)$ with $s(\theta^\star) = 1$, which is the starting point for the next improvement step.

If our assumption holds that the black-box return R really only depends on θ, then Algorithm 1 converges after the second iteration when the optimization step finishes. This follows from the property that the optimization fixes $\theta^\star$ with a constraint, which means that the reinforcement learning has to be executed only once. In practice, this convergence property is not always the case since it strongly depends how R is chosen. To gain more robustness we may perform multiple iterations. However, we will show in the experimental section that our assumption holds for different synthetic and real world problems.

V. CORL for Robotic Manipulation Skills

So far we described our algorithm CORL independent of a specific application domain. The CORL framework allows us to express priors, e.g., in terms of the choice of the policy parametrizations, the objective functions and especially the projection constraint $h(w, \theta)$. In this paper, we will focus on robot manipulation skills and propose such domain knowledge that fits in the framework of Algorithm 1.

A. Policy Parametrization w

We represent the policy as a unique LQR around a reference trajectory $\xi(w,t)$ with $t \in [0, T]$. We assume a linear parametrization of the reference trajectory $\xi(w,t) = B(t)w$ by the policy parameters w. $B(t)$ are RBFs or B-spline basis functions, leading to $n = PQ$ parameters for P basis functions and a Q-dimensional robot configuration.

B. Assumed Projection Constraint $h(w, \theta)$ for Manipulations

A key part of our approach is the lower dimensional representation θ and the corresponding projection constraint $h(w, \theta)$. The choice of these elements requires domain knowledge. In this paper we propose a definition of θ and $h(w, \theta)$ for robot manipulations under the assumption of a rigid body kinematic world. In this case, manipulation means to articulate an external degree of freedom by establishing contact with the respective body. We assume that the dynamics and objectives of the pre-contact motion as well as the post-contact motion of the external degree of freedom can be modeled, while the choice of how to contact the object is a crucial black-box for success, implying parameters θ. In other words, the parts of the motion where the robot is performing the contact are difficult to model with an analytic cost function J, but very important for achieving task success. Following this heuristic, we use the contact points during the manipulation task as lower dimensional representation of θ. If $\phi_{\text{CP}}(\xi(w,t_C))$ is the forward kinematics of the robot's contact points at the time of contact t_c, the projection constraint can be defined as

$$h(w, \theta) = \phi_{\text{CP}}(\xi(w, t_C)) - \theta \ . \tag{6}$$

For example, in a door opening task θ could be the point where the robot is grasping the door. This concept is transferable to other manipulation tasks where the contact points are crucial for performance and success (e.g., button pushing, drawer opening). More generally, θ should capture essential parameters of the interaction with the objects, e.g., where and how to establish contact and where to release contact. Essentially our framework assumes that this interaction parameter space is much lower dimensional than the full robot motion.

In the following, first the motion optimization (Section VI) and afterwards the Bayesian optimization (Section VII) are described in detail.

VI. Motion optimization for fixed θ

In the first part of Algorithm 1 the goal is to improve the policy while keeping the lower dimensional parameter θ fixed. We use a Gauss-Newton motion optimization framework [27] that improves $\xi(w,t)$ w.r.t. J by computing Newton steps. We adapt this framework for learning manipulation tasks in a stepwise manner.

A. Trust-region Policy Improvement

The goal of trajectory optimization is to find optimal parameters w of a trajectory $\xi(w,t)$ that minimize the objective function $J(w)$. We discretize the trajectory $\xi(w,t)$ with a stepsize of Δ_t into $K+1$ points $(x_0, x_1, \dots, x_K)$ where $x_t = \xi(w, t\Delta_t)$. Our objective function is

$$J(w) = \sum_{t=0}^{K} v_t^\top \phi_t^2(\tilde{x}_t) \ . \tag{7}$$

This defines the objective as a weighted sum over all time steps where the costs are defined in form of squared features ϕ. Each cost term depends on a k-order tuple of consecutive states $\tilde{x}_t = (x_{t-k}, \dots, x_{t-1}, x_t)$, containing the current and k previous robot configurations [27]. In addition to this task cost we also consider the projection constraint $h(w,\theta)$ as an equality constraint that can be also defined with features ϕ.

The resulting optimization problem is

$$\begin{aligned} w^{(i+1)} = & \operatorname*{argmin}_{w} J(w) \\ \text{s.t.} \quad & h(w,\theta) = 0, \quad \|w - w^{(i)}\| < \varepsilon_{\max} \end{aligned} \tag{8}$$

where we also added an inequality constraint to limit the stepsize of the trajectories between two iterations with $\varepsilon_{\max}$. This *trust-region* constraint guarantees that the current trajectory is close to the previous trajectory $w^{(i)}$. We could solve this trajectory optimization problem in Equation 8 also without this trust-region constraint until it converges to a fixed-point solution. However, this would lead to potential large steps between different trajectory candidates. This could be problematic since it is not guaranteed that the found trajectory leads to task success. Note also the strong relation to relative entropy policy search which limits the divergence between the trajectory *distribution* before and after the update [18].

We incorporate the constraints with the augmented Lagrangian method and solve the resulting problem with Gauss-Newton optimization [17]. Thereby, we exploit the structure of the gradient and Hessian for efficient optimization (see [27] for more details).

B. Optimizing Smoothness and Phase Profile

Our first objective criteria is smoothness. In our experiments we define configuration space acceleration features

$$\phi_t(\tilde{x}_t) = (x_t - 2x_{t-1} + x_{t-2})/\Delta_t^2 \tag{9}$$

that contribute to the objective (7). We use this feature to select the next policy parameters $w^{(i+1)}$ by minimizing the problem defined in Equation (8).

To achieve smoother motions also w.r.t. the time course of the constraints (e.g., when contacts are established), we additionally optimize the phase of the trajectory and keep the geometry fixed. We optimize the phase profile $p(t) : [0,T] \to [0,1]$ of this trajectory w.r.t. transition costs. We discretize $p(t)$ in $K+1$ points $\hat{p} = [p_0, p_1, \dots, p_K]$ that we use as additional trajectory parameters with the boundary conditions $p_0 = 0$ and $p_K = 1$.

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

$$\begin{aligned} J(\hat{p}) = \sum_{i=0}^{K} & \left((\xi(w, p_{i-1}T) - 2\xi(w, p_i T) + \xi(w, p_{i+1}T))/\Delta_t^2 \right)^2 \\ & + (p_{i-1} - 2p_i + p_{i+1})^2 \, . \end{aligned} \tag{10}$$

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 with parameter w. Similar to the problem in Equation (8) we also limit the stepsize with $\varepsilon_{\max}$.

There are different ways to combine the trajectory improvement and phase improvement with each other. In our experimental evaluation we first apply the trajectory improvement steps until convergence, and afterwards apply the phase improvement steps. Alternatively, one could also iterate between the two steps. It is also possible to extend the motion optimization part with other improvement steps that fit in the Gauss-Newton framework.

VII. Reinforcement learning over θ with unknown Success Constraints

We introduce a black-box reinforcement learning method to improve the policy with respect to the lower dimensional parameter θ. The goal of this improvement strategy is to optimize the black-box return function $R(\theta)$ under the success constraint $S(\theta)$ so as to have a low amount of negative interactions with the system. We use Bayesian optimization to learn a binary classifier for the success constraint $S(\theta)$ and a regression model for the return function $R(\theta)$. We propose a new acquisition function $a(\theta)$ that combines both models in such a way that the next policy is selected in a secure and data-efficient manner.

We briefly introduce some background on Gaussian processes and Bayesian optimization before introducing our reinforcement learning strategy.

A. Background on Gaussian Processes

For both function approximations we use Gaussian processes (GP). The advantage of GPs is that they can express a broad range of different functions and that they provide probability distributions over predictions. A GP defines a probability distribution over functions [19]. We will first handle the regression and afterwards the classification case.

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

$$\mu(\theta_\star) = m(\theta_\star) + k_\star^\top (K + \sigma^2 I)^{-1} r \tag{11}$$

$$\mathbb{V}(\theta_\star) = k(\theta_\star, \theta_\star) - k_\star^\top (K + \sigma^2 I)^{-1} k_\star \tag{12}$$

with $k_\star = k(\Omega, \theta_\star)$, Gram matrix K with $K_{ij} = k(\theta^{(i)}, \theta^{(j)})$, and training inputs $\Omega = [\theta^{(1)}, \ldots, \theta^{(d)}]$ with corresponding targets $r = [r^{(1)}, \ldots, r^{(d)}]^\top$.

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

$$p(s = 1|\theta) = \sigma(g(\theta)). \tag{13}$$

Since this likelihood is non-Gaussian the exact posterior over g is not a Gaussian process—one instead uses a Laplace approximation [16]. For more details regarding GPs we refer to [19].

B. Background on Bayesian Optimization

Bayesian optimization [15] is a strategy to find the maximum of an objective function $R(\theta)$ with $\theta \in \mathbb{R}^m$, where the function $R(\theta)$ is not known in closed-form expression and only noisy observations r of the function value can be made at sampled values θ. These samples are collected in a dataset $\{\theta^{(i)}, r^{(i)}\}_{i=1}^{d}$ that is used to build a GP model of R. The next sample point $\theta^{(d+1)}$ is chosen by maximizing an acquisition function $a(\theta)$. There are many different ways to define this acquisition function [3]. One widely used acquisition function is the *probability of improvement* [14] that is defined as

$$\mathrm{PI}_f(\theta) = P\left(R(\theta) \geq R(\theta^+)\right) = \Phi\left(\frac{\mu(\theta) - R(\theta^+)}{\sqrt{\mathbb{V}(\theta)}}\right) \tag{14}$$

$$\text{with } \theta^+ = \underset{\theta \in \{\theta_1, \ldots, \theta_d\}}{\operatorname{argmax}} R(\theta) \tag{15}$$

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

C. Reinforcement Learning over θ

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

For both GPs we use a squared exponential kernel function

$$\mathrm{k}(\theta, \theta') = \sigma_{\mathrm{sf}}^2 \exp\left(-\tfrac{1}{2}(\theta - \theta')^\top \Lambda^{-1} (\theta - \theta')\right) \tag{16}$$

where $\Lambda = \mathrm{diag}([l_1^2, l_2^2, \ldots, l_D^2])$ 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 $m(x) = c$ to incorporate knowledge that regions where no data points are available yet the unfeasible class is predicted. Therefore we select a constant c smaller than 0 that allows to keep the exploration close to the region where data points are available. We use g_R and g_S to define an acquisition function that we use to select the next sample $\theta^{(d+1)}$.

To select the next data point we introduce the acquisition function

$$a_{\mathrm{PIBU}}(\theta) = \mathrm{PI}_{g_R}(\theta)\ [g_S(\theta) \geq 0] + \mathbb{V}_{g_S}(\theta)\ [g_S(\theta) = 0] \tag{17}$$

that combines the probability of improvement with a boundary uncertainty criteria (PIBU). In Equation (17) $[\cdot]$ denotes the indicator function. The first term describes the probability of improvement (cf. Equation (14)) of g_R in the inner region of the classifier g_S. The second term is the predictive variance of the GP 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.

After selecting the next sample $\theta^{(\mathrm{j}+1)}$ by maximizing Equation (17), we compute the trajectory parameter

$$w^{(i+1)} = \underset{w}{\operatorname{argmin}}\ ||w^\star - w||^2 \quad \text{s.t.} \quad h(w, \theta^{(\mathrm{d}+1)}) = 0 \tag{18}$$

where $w^\star$ is the solution of the previous motion optimization method (see Algorithm 1). The trajectory $\xi(w^{(i+1)}, t)$ is executed on the system and the observed return and task outcome are added to the dataset D. This procedure is repeated until convergence.

VIII. Experiments

We evaluate our approach in two experiments. The first experiment is a synthetic problem where we compare our proposed CORL algorithm with alternative methods. The second experiment is on a PR2 where we optimize the smoothness and interaction forces for the task of opening a door.

Method	**Global optima found**	**Number of failures**	**Max distance to safe region**
PIBU	**100/100**	8.49 ± 1.35	0.71 ± 0.44
PoWER	67/100	8.36 ± 5.91	1.34 ± 0.43
UCB	97/100	14.53 ± 1.07	1.48 ± 0.11
CMA	77/100	7.74 ± 4.41	1.28 ± 0.35
CORL + PIBU	**100/100**	$\mathbf{2.04 \pm 0.19}$	$\mathbf{0.10 \pm 0.04}$
CORL + UCB	92/100	4.38 ± 0.98	1.25 ± 0.69
CORL + CMA	85/100	2.98 ± 2.58	1.26 ± 1.15
CORL + SAL	99/100	$\mathbf{1.74 \pm 0.56}$	$\mathbf{0.06 \pm 0.11}$

Fig. 1: **Evaluation of CORL** (see Section VIII-A)

A. Evaluation of CORL on a Synthetic Benchmark

In this evaluation we consider a synthetic benchmark problem to compare existing black-box RL algorithms with combinations of our CORL framework, including our proposed PIBU aquisition function. We define the synthetic problem in the form of Equation (5) with parameters $w \in \mathbb{R}^2$ and $\theta \in \mathbb{R}$. The problem we optimize is defined by

- an analytic cost $J(w) = (w_1^2 + w_2^2 - 1)^2$,
- a black-box return $R(\theta) = -0.5\theta^2 + \cos(3\theta)$,
- a black-box success $S(\theta) = [-1.5 < \theta < 2.5]$,
- and a projection constraint $h(w, \theta) = \theta - \operatorname{atan}\left(\frac{w_1}{w_2}\right)$.

The total objective we want to minimize is $J(w) - R(\theta)$ under the constraint that $S(\theta) = 1$ (see Equation (5)). We limit the search space to the region $w \in [-1.5, 1.5] \times [-1.5, 1.5]$. This problem has multiple local optima and a global optimum at $w^\star = (1, 0)$ with a value of -1.

We compare two different type of algorithms with each other. The first type uses the CORL framework we proposed in this paper with different black-box reinforcement learning algorithms (noted as CORL + <black-box-method>). The second type are standard reinforcement learning methods that require a single reward objective and ignore the generalized RL structure. To make this comparison possible we therefore define such a single reward objectives as $o(w) = (J(w) - R(\theta))\ [S(\theta) = 1] + 15\ [S(\theta) = 0]$. Consistent to our framework, the return and cost function can only be observed for parameters that lead to success. For failures a constant cost of 15 is received.

The evaluated algorithms are:

- **PIBU:** Bayesian optimization with PIBU (Equation (17))
- **CMA:** Covariance matrix adaptation [9] algorithm
- **UCB:** Bayesian optimization with the acquisition function upper confidence bound (UCB) [3]
- **PoWER:** Policy search algorithm [11]
- **CORL + PIBU:** Algorithm 1 with PIBU (Equation (17))
- **CORL + UCB:** Algorithm 1 with UCB
- **CORL + SAL:** Algorithm 1 with SAL [22]
- **CORL + CMA:** Algorithm 1 with CMA

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

The optimization step in CORL is done with a Newton-Algorithm with a maximum step size $\varepsilon_{\max} = 0.001$. Our PIBU acquisition function used for the GP g_R the hyperparameter $l = 0.4$, $\sigma_{sf} = 10$ and $\sigma = 0.11$. For the classification GP g_S we set $l = 0.4$, $\sigma_{sf} = 10$ and a constant prior mean of -7. We executed all algorithms on this problem from 100 different initial parameters. The initial parameter are sampled uniformly in the search region and fulfill the success constraint.

The table in Figure 1 shows the results over this experiment. We compare the metrics:

- **Global optima found:** This metric describes how many times the algorithm found the global optima $w^\star$.
- **Number of failures:** The number of parameters selected by the algorithm that led to failure $S(\theta) = 0$. All values are given by mean and standard deviation.
- **Max distance to safe region:** The maximum distance between all failure samples to the safety region. All values are given by mean and standard deviation.

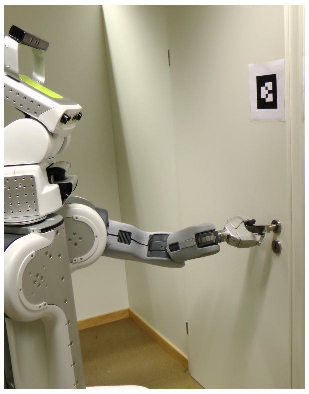
The best two algorithms are marked bold in the table in Figure 1. The CORL + SAL method is as expected the safest method with a mean of 1.74 failure samples—but recall that it assumes it can observe the distance to the boundary in critical regions, which ours does not. Our proposed methods CORL + PIBU and PIBU always find the optimal solution and exhibits a very low number of near-boundary failures even without observing critical distance. The methods that do not take safety into account reach higher number of failure samples (between 5 and 10) that are also located far away from the safety region.

B. Opening a Door with a PR2

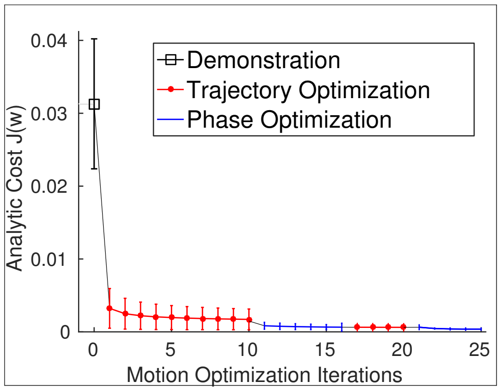
In this experiment we address the task of opening a door. Such kind of manipulation tasks are of special interest since for the part of the motion where the robot is moving freely good models are available but for the part where the robot interacts with objects it is hard to obtain good models. This results from the fact that the environment is usually not completely known (e.g., object position, kinematic structure, physical entities) and that knowledge about how the environment can be manipulated into a certain state is not available.

The task setup is shown in Figure 2a. Starting from a demonstration via kinesthetic teaching the robot improves the task as much as possible. As analytic cost function $J(w)$ we use the sum of squared acceleration in configuration space of the motion with the goal to reach a smooth motion. As black-box return $R(\theta)$ we use the amount of force to open the door measured with a force torque sensor at the robot wrist. The success criteria $S(\theta)$ is a Boolean function that tells if the door was opened successfully. For achieving an autonomous learning we used markers on the door to measure if the door is open and added a simple motion that closes the door after each trial. This allowed the robot to perform the learning on its own without human intervention.

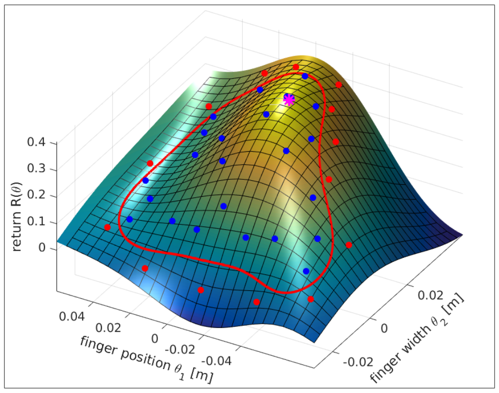
The reference trajectory $\xi(w,t)$ is parametrized by a B-spline with 150 knot points in a 7 dimensional configuration space that leads to 1057 parameter for the full policy w. As lower dimensional parameter θ we define two parameters in



(a)



(b)



(c)

Fig. 2: **Door opening experiments** (see Section VIII-B). The image in (a) shows the task setup of the PR2 opening a door. The learning curve in (b) shows the improvement on the analytic cost function $J(w)$ of the first step of Algorithm 1. The graph in (c) shows the learned return function $R(\theta)$ 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.

the contact space of the door handle. The first parameter is the finger position on the handle relative to the demonstration. The second parameter is the finger opening widths. The parameters for the regression GP g_R we used are $l = 0.0417$, $\sigma_{sf} = 0.1682$ and $\sigma = 0.0120$. For the classification GP g_S we set $l = 0.02$, $\sigma_{sf} = 10$ and a constant prior mean of -7. As stopping criteria for Algorithm 1 we set $\varepsilon_\theta = 0.003$ and $\varepsilon_w = 10^{-5}$.

The results of the motion optimization are shown in Figure 2b. This figure shows how the motion optimization method improves the squared acceleration of the trajectory with the trust-region motion optimization methods (see Section VI). Starting with a demonstration, first the trajectory optimization is applied until convergence and afterwards the phase optimization is applied until convergence. Between iteration 16 and 17 the reinforcement learning with respect to θ was performed and converged after 40 iterations. Afterwards another round of motion optimization is applied until the change of policy parameters was below the threshold. The results of the reinforcement learning in the lower dimensional parameter space θ are shown in Figure 2c. The 40 rollouts of the black-box Bayesian optimization consisted of 26 successes and 14 failures. The blue dots are successful rollouts, the red dots are failures and the magenta star shows the best parameter. The red line denotes the classifier boundary. The demonstration and learning process are shown in a video https://www.youtube.com/watch?v=bn_sv5A1BhQ.

We compared our proposed Bayesian optimization strategy PIBU to CMA. CMA has been used previously to learn robot skills [4, 20, 23]. We applied both methods on the same return function $R(\theta)$ over 100 iterations. The results are shown in the table in Figure 3. We use as performance measure the highest return achieved during the 100 iterations, 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 reaches a lower failure rate with a very low standard deviation. The failures of PIBU are also close to the safety region than CMA. This results from the fact that the boundary is explored with our acquisition function (see Equation (17)). CORL + CMA also finds a slightly worse policy than CORL. Both our method and CMA operate on the low-dimensional parameterization θ that we proposed with the CORL framework, exploiting the combination with the analytic motion optimization.

Method	**Highest return**	**Failure rate**	**Max distance to safe region**
CORL + PIBU	0.45 ± 0.032	0.157 ± 0.021	0.0146 ± 0.006
CORL + CMA	0.41 ± 0.017	0.190 ± 0.102	0.0358 ± 0.030

Fig. 3: Comparison between CORL+CMA and CORL+PIBU

We tried other approaches that do not rely on this low-dimensional projection and the combination with an analytic motion optimizer: We performed experiments with DMP and PoWER similar to [11]. For this we parameterized the shape and goal of the DMP, 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 informative enough for this large parameter space. This reinforces the motivation for our general approach of dissecting returns into high-dimensional analytical and lower dimensional black-box parts.

IX. Conclusion

In this paper we introduced an approach to learning manipulation skills using a structured reinforcement learning formulation. We proposed an algorithm that combines high-dimensional analytic motion optimization and low-dimensional black-box Bayesian optimization. A limitation of our approach is that the choice of h is non-trivial and requires domain knowledge. We discussed a way of choosing h for manipulation tasks where we assumed that the contact points are fixed on the object. An extension for sliding contacts is still an open research question. Another limitation is the generalization ability of the current approach. So far we focused on a trajectory representation for a single scenario. In future research we will investigate the learning of more general skill representations (e.g., cost function) by combining it with inverse optimal control [5].

References

[1] Felix Berkenkamp and Angela P Schoellig. Safe and robust learning control with Gaussian processes. In *Proceedings of European Control Conference*, 2015.

[2] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2001.

[3] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599*, 2010.

[4] Andreas Doerr, Nathan Ratliff, Jeannette Bohg, Marc Toussaint, and Stefan Schaal. Direct loss minimization inverse optimal control. In *Proceedings of Robotics: Science and Systems*, 2015.

[5] Peter Englert and Marc Toussaint. Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations. In *Proceedings of the International Symposium of Robotics Research*, 2015.

[6] Jacob Gardner, Matt Kusner, Zhixiang Xu, Kilian Weinberger, and John Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of International Conference on Machine Learning*, 2014.

[7] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. In *Uncertainty in Artificial Intelligence*, 2014.

[8] Robert B Gramacy and Herbie K H Lee. Optimization under unknown constraints. In *Bayesian Statistics*, volume 9, page 229. Oxford University Press, 2011.

[9] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[10] Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, and Stefan Schaal. Learning force control policies for compliant manipulation. In *International Conference on Intelligent Robots and Systems*, 2011.

[11] Jens Kober and Jan Peters. Policy Search for Motor Primitives in Robotics. *Machine Learning*, pages 171–203, 2011.

[12] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 2013.

[13] Andras G Kupcsik, Marc P Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the National Conference on Artificial Intelligence*, 2013.

[14] Harold J Kushner. 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, 1964.

[15] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129), 1978.

[16] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(10), 2008.

[17] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

[18] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the Conference on Artificial Intelligence*, 2010.

[19] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[20] Elmar A Rückert, Gerhard Neumann, Marc Toussaint, and Wolfgang Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience*, 6, 2013.

[21] Matthias Schonlau, William J Welch, and Donald R Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, pages 11–25, 1998.

[22] Jens Schreiter, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint. Safe Exploration for Active Learning with Gaussian Processes. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2015.

[23] Freek Stulp and Olivier Sigaud. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics*, 4(1):49–61, 2013.

[24] Freek Stulp, Olivier Sigaud, and Others. Policy improvement methods: Between black-box optimization and episodic reinforcement learning. *Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes*, 2013.

[25] Yanan Sui, Alkis Gotovos, Joel W Burdick, and Andreas Krause. Safe exploration for optimization with Gaussian processes. In *Proceedings of International Conference on Machine Learning*, 2015.

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

[27] Marc Toussaint. Newton methods for k-order Markov constrained motion problems. *arXiv:1407.0414 [cs.RO]*, 2014.

[28] Rok Vuga, Bojan Nemec, and Ales Ude. Enhanced policy adaptation through directed explorative learning. *International Journal of Humanoid Robotics*, 12(03), 2015.