Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations

Englert, P., Vien, N. A., & Toussaint, M. (2017). Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations. *International Journal of Robotics Research*, *36*(13-14), 1474-1488. https://doi.org/10.1177/0278364917745980

Published in:
International Journal of Robotics Research

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations

Peter Englert[1], Ngo Anh Vien[2] and Marc Toussaint[1]

Abstract

Inverse Optimal Control (IOC) assumes that demonstrations are the solution to an optimal control problem with unknown underlying costs, and extracts parameters of these underlying costs. We propose the framework of Inverse KKT, which assumes that the demonstrations fulfill the Karush-Kuhn-Tucker conditions of an unknown underlying constrained optimization problem, and extracts parameters of this underlying problem. Using this we can exploit the latter to extract the relevant task spaces and parameters of a cost function for skills that involve contacts. For a typical linear parameterization of cost functions this reduces to a quadratic program, ensuring guaranteed and very efficient convergence, but we can deal also with arbitrary non-linear parameterizations of cost functions. We also present a nonparametric variant of inverse KKT that represents the cost function as a functional in reproducing kernel Hilbert spaces. The aim of our approach is to push learning from demonstration to more complex manipulation scenarios that include the interaction with objects and therefore the realization of contacts/constraints within the motion. We demonstrate the approach on manipulation tasks such as sliding a box, closing a drawer and opening a door.

Keywords

Imitation Learning, Inverse Optimal Control, Manipulation Skills

1 Introduction

Most tasks in real world scenarios require contacts with the environment. For example, the task of opening a door requires contact between the robot gripper and the door handle. In this paper we address learning from demonstration for the case of manipulation that incorporates contacts. Specifically, we want to

[1]Machine Learning & Robotics Lab, Universität Stuttgart, Germany
[2]School of EEECS, Queen's University Belfast, UK

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

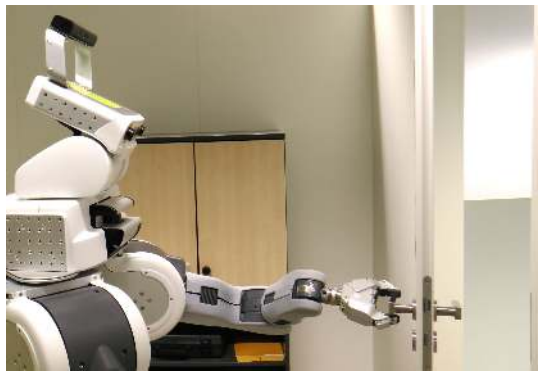


Figure 1. This picture shows the door opening task that we use to evaluate our approach by first learning a cost function from demonstration and then generating motions by optimizing the cost function for different scenarios (e.g., initial positions, door angles). See Section 6.5 for more details.

extract from demonstrations how to represent and execute manipulations in such a way that the robot can perform such tasks in a robust and general manner.

Cost functions are a powerful representation for robot skills, since they are able to encode task knowledge in a very abstract way. This property allows them to reach high generalization to a wide range of problem configurations. However, designing cost functions by hand can be hard since the right features have to be chosen and combined with each other. Therefore, inverse optimal control, also known as inverse reinforcement learning (Ng and Russell (2000)), tries to automate the design of cost functions by extracting the important task spaces and cost parameters from demonstrations. 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)).

There are two parts necessary for applying learning from demonstration with IOC: 1) The inverse optimization method for extracting the cost function from demonstrations; 2) The motion optimization method that creates motions by minimizing such cost functions. Both parts are coupled by the cost function, which is the output of the first and input of the second part, see Figure 2. Usually IOC algorithms try to find a cost function such that the output of the motion optimization method is similar to the input demonstrations of the inverse problem. Therefore, the cost function is used as a compact representation that encodes the demonstrated behavior.

Our approach finds a cost function, including the identification of relevant task spaces, such that the demonstrations fulfill the KKT conditions of an underlying constrained optimization problem with this cost function. Thereby we integrate constraints into the IOC method, which allows us to learn from object manipulation demonstrations that naturally involve contact constraints. Motion generation for such cost functions (point 2 above) is a non-linear constrained program which we solve using an augmented Lagrangian method. However, for typical cost function parameterizations, the IOC problem of inferring the cost function parameters (point 1 above) becomes a quadratic program, which can be solved very efficiently.

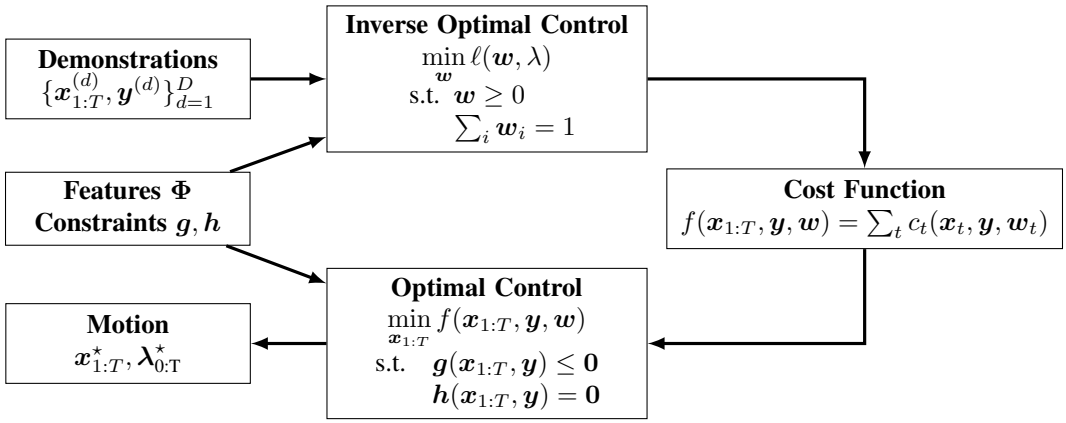


Figure 2. Concept of skill learning with inverse optimal control, where the cost function plays the central role of encoding the demonstrated behavior. In this paper, we present our formulation of learning a cost function for a constrained trajectory optimization problem.

The structure of the paper is as follows. We would like to defer the discussion of related work to after we have introduced our method, in Section 5. In Section 2, we introduce some background on constrained trajectory optimization, which represents the counterpart to the IOC approach. We develop our IOC algorithm in Section 3 by deriving a cost function for the inverse problem based on KKT conditions. In Section 4 we present a nonparametric variant of inverse KKT that. In Section 6, we evaluate our approach on simulated and real robot experiments.

The inverse KKT formulation was initially presented in (Englert and Toussaint 2015). The main contribution is the formulation of an IOC method for constrained motions with equality and inequality constraints that is based on the KKT conditions. This method allows to efficiently extract task spaces and parameters of a cost function from demonstrations.

2 Constrained Trajectory Optimization

A trajectory $\boldsymbol{x}_{0:T}$ is a sequence of $T+1$ robot configurations $\boldsymbol{x}_t \in \mathbb{R}^n$. The goal of trajectory optimization is to find a trajectory $\boldsymbol{x}_{1:T}^{\star}$, given an initial configuration $\boldsymbol{x}_0$, that minimizes a certain objective function

$$f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = \sum_{t=1}^{T} c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t) \ . \tag{1}$$

This defines the objective as a sum over cost terms $c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t)$, where each cost term depends on a k-order tuple of consecutive states $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$, containing the current and k previous robot configurations (Toussaint (2017)). This allows us to specify costs on the level of positions, velocities or accelerations (for $k = 2$) in configuration space as well as any task spaces. In addition to the robot configuration state $\tilde{\boldsymbol{x}}_t$ we use external parameters of the environment $\boldsymbol{y}$ to contain information that are

important for planning the motion (parameters of the environment's configuration, e.g. object positions). These $\boldsymbol{y}$ usually vary between different problem instances, which is used to generalize the skill to different environment configurations.

We typically assume that the cost terms in Equation (1) are a weighted sum of squared features,

$$c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t) = \boldsymbol{w}_t^\top \boldsymbol{\phi}_t^2(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) \;, \tag{2}$$

where $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ are the features and $\boldsymbol{w}_t$ is the weighting vector at time t. A simple example for a feature is the robot's end-effector position at the end of the motion T relative to the position of an object. In this example the feature $\boldsymbol{\phi}_T(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ would compute the difference between the forward kinematics mapping and object position (given by $\boldsymbol{y}$). More complex tasks define body orientations or relative positions between robot and an object. Transition costs are a special type of features, which could be squared torques, squared accelerations or a combination of those, or velocities or accelerations in any task space.

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 are analogous to features $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ and can refer to arbitrary task spaces. An example for an inequality constraint is the distance to an obstacle, which should not be below a certain threshold. In this example $\boldsymbol{g}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ would be the smallest difference between the distance of the robot body to the obstacle and the allowed threshold. The equality constraints are in our approach mostly used to represent persistent contacts with the environment (e.g., $\boldsymbol{h}_t$ 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 (2), is the fact that minimizing costs does not guarantee that they will become $\boldsymbol{0}$, which is essential for establishing a contact.

For better readability we transform Equation (1) and Equation (3) into vector notation by introducing the vectors $\boldsymbol{w}$, $\boldsymbol{\Phi}$, $\boldsymbol{g}$ and $\boldsymbol{h}$ that concatenate all elements over time. This allows us to write the objective function of Equation (1) as

$$f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = \boldsymbol{w}^\top \Phi^2(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \tag{4}$$

and the overall optimization problem as

$$\begin{aligned} \boldsymbol{x}_{1:T}^\star = \arg\min_{\boldsymbol{x}_{1:T}} \; & f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) = \boldsymbol{0} \end{aligned} \tag{5}$$

We solve such problems using the augmented Lagrangian method (Nocedal and Wright (2006)). Therefore, additionally to the solution $\boldsymbol{x}_{1:T}^\star$ we also get the Lagrange parameters $\boldsymbol{\lambda}_{1:T}^\star$, 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 (Toussaint et al. (2014)). We use a Gauss-Newton optimization method to solve the unconstrained Lagrangian problem in the inner loop of augmented Lagrangian. For this, the gradient is

$$\nabla_{\boldsymbol{x}_{1:T}} f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = 2\boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y})^\top \text{diag}(\boldsymbol{w}) \Phi(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \tag{6}$$

and the Hessian is approximated as in Gauss-Newton as

$$\nabla^2_{\boldsymbol{x}_{1:T}} f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) \approx 2\boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y})^\top \text{diag}(\boldsymbol{w}) \boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y}), \tag{7}$$

where $\boldsymbol{J} = \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{x}}$ is the Jacobian of the features. Using a gradient based trajectory optimization method restricts the class of possible features $\boldsymbol{\Phi}$ to functions that are continuous with respect to $\boldsymbol{x}$. However, we will show in the experimental section that this restriction still allows to represent complex behavior like opening a door or sliding a box on a table.

3 Inverse KKT Motion Optimization

We now present the inverse KKT method (Englert and Toussaint 2015), which is a way to solve the inverse problem for the constrained trajectory optimization formulation introduced in the previous section. We assume that D demonstrations of a task are provided with the robot body (e.g., through teleoperation or kinesthetic teaching) and are given in the form $(\hat{\boldsymbol{x}}^{(d)}_{0:T}, \hat{\boldsymbol{y}}^{(d)})_{d=1}^{D}$, where $\hat{\boldsymbol{x}}^{(d)}_{0:T}$ is the demonstrated trajectory and $\hat{\boldsymbol{y}}^{(d)}$ is the environment configuration (e.g., object position). Another assumption we make is that the constraints $\boldsymbol{g}$ and $\boldsymbol{h}$ and a set of potential features $\boldsymbol{\Phi}$ are provided as input. Inverse KKT learns the weight vector $\boldsymbol{w}$ of these features from the demonstrations.

3.1 Inverse KKT Objective

Our IOC objective is derived from the Lagrange function of the problem in Equation (5)

$$L(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) + \boldsymbol{\lambda}^\top \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \\ \boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \end{bmatrix} \tag{8}$$

and the Karush-Kuhn-Tucker (KKT) conditions. The first KKT condition says that for an optimal solution $\boldsymbol{x}^\star_{1:T}$ the condition

$$\nabla_{\boldsymbol{x}_{1:T}} L(\boldsymbol{x}^\star_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = \boldsymbol{0} \tag{9}$$

has to be fulfilled. With Equation (6) this leads to

$$2\boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y})^\top \text{diag}(\boldsymbol{w}) \Phi(\boldsymbol{x}_{1:T}, \boldsymbol{y}) + \boldsymbol{J}_c(\boldsymbol{x}_{1:T}, \boldsymbol{y})^\top \boldsymbol{\lambda} = \boldsymbol{0} \tag{10}$$

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

We express this idea in terms of the loss function

$$\ell(\boldsymbol{w}, \boldsymbol{\lambda}) = \sum_{d=1}^{D} \ell^{(d)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(d)}) \tag{11}$$

with

$$\ell^{(d)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(d)}) = \left|\left| \nabla_{\boldsymbol{x}_{1:T}} L(\hat{\boldsymbol{x}}^{(d)}_{0:T}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\lambda}^{(d)}, \boldsymbol{w}) \right|\right|^2 , \tag{12}$$

where we sum over D demonstrations of the scalar product of the first KKT condition. In Equation (11), d enumerates the demonstrations and $\boldsymbol{\lambda}^{(d)}$ is the dual to the demonstration $\hat{\boldsymbol{x}}^{(d)}_{0:T}$ 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, $\boldsymbol{\lambda}^{(d)} = \boldsymbol{\lambda}^{(d)}(\hat{\boldsymbol{x}}^{(d)}_{0:T}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{w})$ is a function of the primal demonstration $\hat{\boldsymbol{x}}^{(d)}_{0:T}$, the environment configuration of that demonstration $\hat{\boldsymbol{y}}^{(d)}$, and the underlying parameters $\boldsymbol{w}$. And $\ell^{(d)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(d)}(\boldsymbol{w})) = \ell^{(d)}(\boldsymbol{w})$ becomes a function of the parameters only (we think of $\hat{\boldsymbol{x}}^{(d)}_{0:T}$ and $\hat{\boldsymbol{y}}^{(d)}$ as given, fixed quantities, as in Equations (11-12)).

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

$$\nabla_{\boldsymbol{\lambda}^{(d)}} \ell^{(d)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(d)}) = \mathbf{0} \tag{13}$$

$$\Rightarrow \quad \boldsymbol{\lambda}^{(d)}(\boldsymbol{w}) = -2(\tilde{\boldsymbol{J}}_{\boldsymbol{c}} \tilde{\boldsymbol{J}}_{\boldsymbol{c}}^{\top})^{-1} \tilde{\boldsymbol{J}}_{\boldsymbol{c}} \boldsymbol{J}^{\top} \mathrm{diag}(\boldsymbol{\Phi}) \boldsymbol{w} \, . \tag{14}$$

Note that here the matrix $\tilde{\boldsymbol{J}}_{\boldsymbol{c}}$ 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 $\boldsymbol{h}$ are independent of $\boldsymbol{w}$. This ensures that (14) is the minimizer subject to the complementarity condition. The number of active constraint at each time point has a limit. This limit would be exceeded if more degrees of freedom of the system are constrained than there are available.

By inserting Equation (14) into Equation (12) we get

$$\ell^{(d)}(\boldsymbol{w}) = 4\boldsymbol{w}^{\top} \underbrace{\mathrm{diag}(\boldsymbol{\Phi}) \boldsymbol{J} \Big(\boldsymbol{I} - \tilde{\boldsymbol{J}}_{\boldsymbol{c}}^{\top} (\tilde{\boldsymbol{J}}_{\boldsymbol{c}} \tilde{\boldsymbol{J}}_{\boldsymbol{c}}^{\top})^{-1} \tilde{\boldsymbol{J}}_{\boldsymbol{c}} \Big) \boldsymbol{J}^{\top} \mathrm{diag}(\boldsymbol{\Phi})}_{\boldsymbol{\Lambda}^{(d)}} \boldsymbol{w} \tag{15}$$

which is the IOC cost per demonstration (see appendix A for a detailed derivation). Adding up the loss per demonstration and plugging this into Equation (11) we get a total inverse KKT loss of

$$\ell(\boldsymbol{w}) = \boldsymbol{w}^{\top} \Lambda \boldsymbol{w} \qquad \text{with} \quad \Lambda = 4 \sum_{d=1}^{D} \Lambda^{(d)}. \tag{16}$$

The resulting optimization problem is

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \boldsymbol{w}^{\top} \Lambda \boldsymbol{w} \\ &\text{s.t.} \quad \boldsymbol{w} \geq \mathbf{0} \end{aligned} \tag{17}$$

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 (4). Our approach also works in the unconstrained case. In this case the constraint term vanishes in Equation (10) and the remaining part is the optimality condition of unconstrained optimization, which says that the gradient of the cost function should be equal to zero.

3.2 Regularization & Sparsity

The above formulation may lead to the singular solution $\boldsymbol{w} = \mathbf{0}$ where zero costs are assigned to all demonstrations, trivially fulfilling the KKT conditions. This calls for a regularization of the problem.

In principle there are two ways to regularize the problem to enforce a non-singular solution: First, we can impose positive-definiteness of Equation (4) at the demonstrations (cf. Levine and Koltun (2012)). Second, as the absolute scaling of Equation (4) is arbitrary we may additionally add the constraint

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \boldsymbol{w}^\top \Lambda \boldsymbol{w} \\ &\text{s.t.} \quad \boldsymbol{w} \geq \mathbf{0} \ , \quad \sum_i \boldsymbol{w}_i = 1 \end{aligned} \tag{18}$$

to our problem formulation (17). We choose the latter option in our experiments. Equation (18) is a (convex) quadratic program (QP), 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.

There exist different ways to modify the problem in Equation (18) such that the solutions become sparse. One possibility is to subtract the regularization term $\boldsymbol{w}^\top \boldsymbol{w}$ from the IOC loss function in Equation (18). Another possibility to achieve sparse solutions is to change the equality constraint into $\sum_i \boldsymbol{w}_i^p = 1$ with a $p > 2$. In this case the problem is not convex anymore.

3.3 Linear & Nonlinear Weight Parametrization

In practice we usually use parametrizations on $\boldsymbol{w}$. This is useful since in the extreme case, when for each time step a different parameter is used, this leads to a very high dimensional parameter space (e.g., 10 tasks and 300 time steps lead to 3000 parameter). This space can be reduced by using the same weight parameter over all time steps or to activate a task only at some 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 IOC method learns. This parametrization allows a flexible assignment of one parameter to multiple task costs. Further linear parametrizations are radial basis function or B-spline basis functions over time t to more compactly describe smoothly varying cost parameters. For such linear parametrization the problem in Equation (18) remains a QP that can be solved very efficiently.

Another option we will consider in the evaluations is to use a nonlinear mapping $\boldsymbol{w}(\boldsymbol{\rho}) = \mathcal{A}(\boldsymbol{\rho})$ to more compactly represent all parameters. For instance, the parameters $\boldsymbol{w}$ can be of a Gaussian shape (as a function of t), where the mean and variance of the Gaussian is described by $\boldsymbol{\rho}$. Such a parametrization would allow us to learn directly the time point when costs are active. In such a case, the problem is not convex anymore. We address such problems using a general non-linear programming method (again, augmented Lagrangian) and multiple restarts are required with different initializations of the parameter.

3.4 Feature & Constraint Design

Our IOC method requires equality constraints $\boldsymbol{h}$, inequality constraints $\boldsymbol{g}$ and a set of potential features $\boldsymbol{\Phi}$ as inputs (see Figure 2). Extracting the features and constraints from the demonstrations is nontrivial. We propose to first define a set of features $\boldsymbol{\Phi}(\boldsymbol{x}_{1:T}, \boldsymbol{y})$ that could be relevant for the task. The subset of $\boldsymbol{\Phi}$ that fulfill the condition

$$\boldsymbol{\phi}(\hat{\boldsymbol{x}}_t^{(1)}, \hat{\boldsymbol{y}}^{(1)}) = \boldsymbol{\phi}(\hat{\boldsymbol{x}}_t^{(2)}, \hat{\boldsymbol{y}}^{(2)}) = \ldots = \boldsymbol{\phi}(\hat{\boldsymbol{x}}_t^{(D)}, \hat{\boldsymbol{y}}^{(D)}) \ . \tag{19}$$

are used as equality constraints $\boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y})$. The remaining features are kept for the cost function. We used the following feature types for the real robot experiments:

- **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.
- **Orientation features:** Represent orientation of a body relative to another body.

A body is either a part of the robot or belongs to an object in the environment. We define these features at different time points that are extracted from the demonstration (e.g., zero velocity, contact release) or learned with a RBF parametrization (see experiment in Section 6.2).

We use the equality constraints $\boldsymbol{h}$ mainly to describe contacts between the robot and the environment since they are crucial for task success. The inequality constraints $\boldsymbol{g}$ are used to incorporate collision avoidance and to ensure the robots joint limits. Additionally, we use the constraints to define reasonable behavior on the interaction with the environment. An equality constraint is used to fix external degrees of freedom (e.g., a door) when they are not being manipulated and an inequality is used to constraint the movement direction of external objects (e.g., pushing in a certain direction).

4 Nonparametric Inverse KKT

In this section, we propose a nonparametric variant of the inverse KKT method. The advantage over the parametric variant is that a kernel function can be used that measures the similarity to the demonstrations and no features have to be constructed by hand. In Section 3, the objective function $f(\boldsymbol{x})$ is represented as a weighted sum of squared features (see Equation (1)). In the nonparametric inverse KKT we represent the objective function as

$$f(\boldsymbol{x}) = \sum_{t=1}^{T} c_t(\boldsymbol{x}_t) \tag{20}$$

where each c_t is a functional in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with a reproducing kernel k. Similar to the previous parametric formulation, we write c as

$$c_t(\boldsymbol{x}) = \sum_{i=1}^{n} w_t^{(i)} \psi_i(\boldsymbol{x}_t) = \boldsymbol{w}_t^\top \boldsymbol{\psi}(\boldsymbol{x}_t) \tag{21}$$

where $\psi_i(\boldsymbol{x}_t) = \phi_i^2(\boldsymbol{x}_t)$. According to the representer theorem (Schölkopf and Smola (2002)), the parameter vector $\boldsymbol{w}_t$ can be represented with the demonstrations as

$$\boldsymbol{w}_t = \sum_{d=1}^{D} \alpha_t^{(d)} \boldsymbol{\psi}(\hat{\boldsymbol{x}}_t^{(d)}) \,. \tag{22}$$

Hence, the function c_t in RKHS can be defined as

$$c_t(\boldsymbol{x}_t) = \sum_{d=1}^{D} \alpha_t^{(d)} \langle \boldsymbol{\psi}(\hat{\boldsymbol{x}}_t^{(d)}), \boldsymbol{\psi}(\boldsymbol{x}_t) \rangle \tag{23}$$

$$= \sum_{d=1}^{D} \alpha_t^{(d)} k(\hat{\boldsymbol{x}}_t^{(d)}, \boldsymbol{x}_t) \tag{24}$$

with a kernel k. This means we can use any kernel to represent our cost function and the search for c_t is equal to directly optimizing α. In the following we will use the RBF kernel

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) \ . \tag{25}$$

Similar to the derivation of the inverse KKT loss function in Equations (11)–(15), we use the KKT conditions to formulate the loss function in the nonparametric case. The gradient of the objective function is

$$\nabla_{\boldsymbol{x}_{1:T}} f = \sum_{t=1}^{T} \nabla_{\boldsymbol{x}_{1:T}} c_t(\boldsymbol{x}_t) \tag{26}$$

$$= \left[\frac{\partial c_1(\boldsymbol{x}_1)}{\partial \boldsymbol{x}_1}, \dots, \frac{\partial c_T(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T}\right]^\top \tag{27}$$

with

$$\frac{c_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} = -\sum_{d=1}^{D} 2\alpha_t^{(d)} k(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t^{(d)})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t^{(d)})\boldsymbol{\Sigma}^{-1} \ . \tag{28}$$

The resulting loss function for a demonstration is

$$\ell^{(d)}(\boldsymbol{\alpha}) = \nabla \boldsymbol{f}_{\boldsymbol{x}_{1:T}}^\top \left(\boldsymbol{I} - \boldsymbol{J}_c^\top (\boldsymbol{J}_c \boldsymbol{J}_c^\top)^{-1} \boldsymbol{J}_c\right) \nabla \boldsymbol{f}_{\boldsymbol{x}_{1:T}} \tag{29}$$

$$= \boldsymbol{\alpha}^\top \boldsymbol{\Omega}^{(d)} \boldsymbol{\alpha} \tag{30}$$

where $\boldsymbol{\Omega}^{(d)}$ contains all the terms that are independent of $\boldsymbol{\alpha}$.

Similar to the previous section, we sum over all demonstrations and add a regularization term that results in the nonparametric IKKT optimization problem

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^\top \boldsymbol{\Omega} \boldsymbol{\alpha} \qquad \text{with} \qquad \boldsymbol{\Omega} = \sum_{d=1}^{D} \boldsymbol{\Omega}^{(d)} \tag{31}$$

$$\text{s.t.} \quad \sum_i \alpha_i = 1 \ . \tag{32}$$

The problem can be optimized very efficiently and leads to an unique solution. A difficulty in this nonparametric case is to find a suitable kernel for the problem with a good choice of hyperparameters ($\boldsymbol{\Sigma}$ in Equation (25)). In practice crossvalidation on a test and training set, hyperparameter learning or multiple kernel learning methods can be used to solve this problem (Gönen and Alpaydın (2011)).

5 Related Work

In the recent years there has been extensive research on imitation learning and inverse optimal control. In the following section we will focus on the approaches and methods that are most related to our work of learning cost functions for manipulation tasks. For a broader overview on IOC approaches we refer the reader to the survey paper of Zhifei and Joo (2012) and for an overview on general imitation learning we recommend Argall et al. (2009).

5.1 Max-Entropy and Lagrangian-Based IOC Approaches

The work of Levine and Koltun (2012) is perhaps the closest to our approach. They use a probabilistic formulation of inverse optimal control that approximates the maximum entropy model (MaxEnt) of Ziebart et al. (2008). Similar to MaxEnt, other approaches such as maximum-margin planning (MMP) and LEARCH (LEArning to seaRCH) of Ratliff et al. (2006, 2009) use forward solvers or policy optimization, e.g. value iteration or A*, in the inner loop which would i) require perfect knowledge of the environment dynamics; and ii) hence consume more computation. In our framework of trajectory optimization (cf. Section 2) this translates to

$$\min_{\boldsymbol{w}} \nabla_{\boldsymbol{x}} f^\top (\nabla^2_{\boldsymbol{x}} f)^{-1} \nabla_{\boldsymbol{x}} f - \log |\nabla^2_{\boldsymbol{x}} f|. \tag{33}$$

The first term of this equation is similar to our loss in Equation (11), where the objective is to get small gradients. Additionally, they use the inverse Hessian as a weighting of the gradient. The second term ensures the positive definiteness of the Hessian and also acts as a regularizer on the weights. The learning procedure is performed by maximizing the log-likelihood of the approximated reward function. Instead of enforcing a fully probabilistic formulation, we focus on finite-horizon *constrained* motion optimization formulation with the benefit that it can handle constraints and leads to a fast QP formulation. Further, our formulation also targets at efficiently extracting the relevant task spaces. which deals better with sub-optimal demonstration and noisy data than our formulation. Maxent is like other Bayesian approaches (Ramachandran and Amir 2007) very robust. However, it is proposed to the simple case of linear dynamics and quadratic rewards (LQR). This is hardly the case of arbitrary trajectory optimization. On the contrary, our formulation is based on constrained trajectory optimization, which learns a cost function that fits well with many trajectory optimization solvers and therefore can deal with a wider range of optimal control problems.

Puydupin-Jamin et al. (2012) introduced an approach to IOC that also handles linear constraints. It learns the weight parameter $\boldsymbol{w}$ and Lagrange parameter $\boldsymbol{\lambda}$ by solving a least-squares optimization problem

$$\min_{\boldsymbol{w},\boldsymbol{\lambda}} \left(\begin{bmatrix} 2\boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{\Phi}) & \boldsymbol{J}_c^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} + \boldsymbol{J}^{/w} \right)^2 \tag{34}$$

where $^{/w}$ denotes the part in the cost function that is not weighted with $\boldsymbol{w}$. The method only addresses equality constraints (no complementarity condition for $\boldsymbol{\lambda}$). Our main concern with this formulation is that there are no constraints that ensure that the weight parameter $\boldsymbol{w}$ do not become 0 or negative. If $\boldsymbol{J}^{/w}$ is zero, as in our case, the solution is identially zero $(\boldsymbol{w}, \boldsymbol{\lambda})$. Starting with the KKT condition, they derive a linear residual function that they optimize analytically as the unconstrained least squares. In the experimental section they consider human locomotion with a unicycle model, where they learn one weight parameter of torques and multiple constraints that define the dynamics of the unicycle model and the initial and target position. The idea of using KKT conditions is similar to our approach. However, our formulation allows for inequality constraints and leads to a QP with boundary constraints that ensures that the resulting parameters are feasible. Instead of optimizing for $\boldsymbol{\lambda}$, we eliminate $\boldsymbol{\lambda}$ from the inverse KKT optimization using Equation (14).

The work of Albrecht et al. (2011) learns cost functions for human reaching motions from demonstrations that are a linear combination of different transition types (e.g., jerk, torque). They transformed a bilevel optimization problem, similar to Mombaur et al. (2010), into a constrained

optimization problem of the form

$$\min_{\boldsymbol{x}_{1:T},\boldsymbol{w},\boldsymbol{\lambda}} \left(\phi^{\text{pos}}(\boldsymbol{x}_T) - \phi^{\text{pos}}(\hat{\boldsymbol{x}}_T^{(d)})\right)^2 \tag{35}$$

$$\text{s.t.} \qquad \nabla_{\boldsymbol{x}_{1:T}} L(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = \mathbf{0} \tag{36}$$

$$\boldsymbol{h}(\boldsymbol{x}_{1:T}) = 0 \qquad \sum_i w_i = 1 \qquad \boldsymbol{w} \geq 0 \tag{37}$$

The objective is the squared distance between optimal and demonstrated final hand position. They optimize this objective for the trajectory $\boldsymbol{x}_{1:T}$, the parameter $\boldsymbol{w}$ and the Lagrange parameter $\boldsymbol{\lambda}$ with the constraints that the KKT conditions of the trajectory $\boldsymbol{x}_{1:T}$ are fulfilled. To apply this approach demonstrations are first preprocessed by extracting a characteristic movement with dynamic time warping and a clustering step. Their results show that a combination of different transition costs represent human arm movements best and that they are able to generalize to new hand positions. The advantage of their approach is that they do not only get the parameter weights $\boldsymbol{w}$, but also an optimal trajectory $\boldsymbol{x}_{1:T}^{\star}$ out of the inverse problem in Equations (35)–(37). The use of the KKT conditions differs from our approach in two ways. First, they use the KKT conditions in the constraint part of the formulation in Equation (36), whereas we use them directly as scalar product in the cost function. Second, they use them on the optimization variables $\boldsymbol{x}_{1:T}$, whereas we use them on the demonstrations $\hat{\boldsymbol{x}}^{(d)}$ (see Equation (11)). Instead of minimizing a function directly of the final end-effector position and only learning weights of transition costs, we present a more general solution to imitation learning that can learn transition and task costs in arbitrary feature spaces. Our approach also handles multiple demonstrations directly without preprocessing them to a characteristic movement.

5.2 Black-box Inverse Optimal Control

Black-box optimization approaches are another category of methods for IOC. There, usually an optimization procedure with two layers is used, where in the outer loop black box optimization methods are used to find suitable parameter of the inner motion problem. For this usually no gradients of the outer loop cost function are required.

Mombaur et al. (2010) use such a two-layered approach, where they use in the outer loop a derivative free trust region optimization technique and in the inner loop a direct multiple shooting technique. The fitness function of their outer loop is the squared distance between inner loop solution and demonstrations. They apply it on human locomotion task where they record demonstration of human locomotion and learn a cost function that they transfer to a humanoid robot. Rückert et al. (2013) uses a similar idea to learn movements. They use covariance matrix adaptation (Hansen and Ostermeier (2001)) in the outer loop to learn policy parameters of a planned movement primitive represented as a cost function. Doerr et al. (2015) propose to do policy search on a reward function that measures similarity do demonstrations. They also use covariance matrix adaptation to learn parameters of a trajectory optimization problem that is similar to our formulation in Equation (5). The advantage of their method is that they can use any parameter in the optimization problem as search parameter and define black-box objectives. However, such methods usually have high computational costs for higher-dimensional spaces since the black box optimizer needs many evaluations. Their experimental evaluation for pointing tasks show that they require between 2000 and 4000 evaluations of the forward problem. One also needs

to find a cost function for the outer loop that leads to reasonable behavior. In our problem formulation we do not require any evaluations of the forward problem and the inverse cost function is given by the KKT conditions. A hierarchical combination of analytic IOC and black-box IOC could also be worth studying, where the analytic method optimizes the linear parameter and the black-box method optimizes the nonlinear parameters of the cost function.

5.3 Task Space Extraction

Jetchev and Toussaint (2014) discover task relevant features by training a specific kind of value function, assuming that demonstrations can be modelled as down-hill walks of this function. Similar to our approach, the function is modelled as linear in several potential task spaces, allowing to extract the one most consistent with demonstrations. In Muhlig et al. (2009) they automatically select relevant task spaces from demonstrations. Therefore, the demonstrations are mapped on a set of predefined task spaces, which is then searched for the task spaces that best represent the movement. In contrast to these methods, our approach more rigorously extracts task dimensions in the inverse KKT motion optimization framework, including motions that involve contacts.

5.4 Model-free Imitation Learning

Another approach is the widely used framework of direct imitation learning with movement primitives (Schaal et al. (2003); Paraschos et al. (2013); Pastor et al. (2011)). They belong to a more direct approach of imitation learning that does not try to estimate the cost function of the demonstration. Instead they represent the demonstrations in a parametrized form that is used to generalize to new situations (e.g., changing duration of motion, adapting the target). Many extensions with different parametrization exist that try to generalize to more complex scenarios (Calinon et al. (2013); Stulp et al. (2013)). They are very efficient to learn from demonstrations and have been used for manipulation tasks (e.g., manipulating a box).

There also exist IOC methods that are model-free (Boularias et al. 2011; Kalakrishnan et al. 2013; Finn et al. 2016). Kalakrishnan et al. (2013) introduce an inverse formulation of the path integral reinforcement learning method PI^2 (Theodorou et al. (2010)) to learn objective functions for manipulation. The cost function consists of a control cost and a general state dependent cost term at each time step. They maximize the trajectory likelihood of demonstrations $p(\hat{\boldsymbol{x}}_{0:T}|\boldsymbol{w})$ for all demonstrations by creating sampled trajectories around the demonstrations. Further, they L1 regularize $\boldsymbol{w}$ to only select a subset of the weights. The method is evaluated on grasping tasks. Finn et al. (2016) propose to learn a cost function in an inner loop of a policy search method. They formulate a sample-based approximation for nonlinear maximum entropy IOC. As cost function representation a neural network is used and regularization is achieved by penalizing an acceleration term and preferring strict monotonically decrease in the costs of the demonstration. They evaluate their method on robot manipulation tasks that include autoencoder features from camera images.

The major difference of such kind of approaches to our method is that they do not need an internal model of the environment, which is sometimes difficult to obtain. However, if such a model is available it can be used to learn a cost function that provide better generalization abilities than movement primitives. This is the case since cost functions are a more abstract representation of task knowledge. Examples of

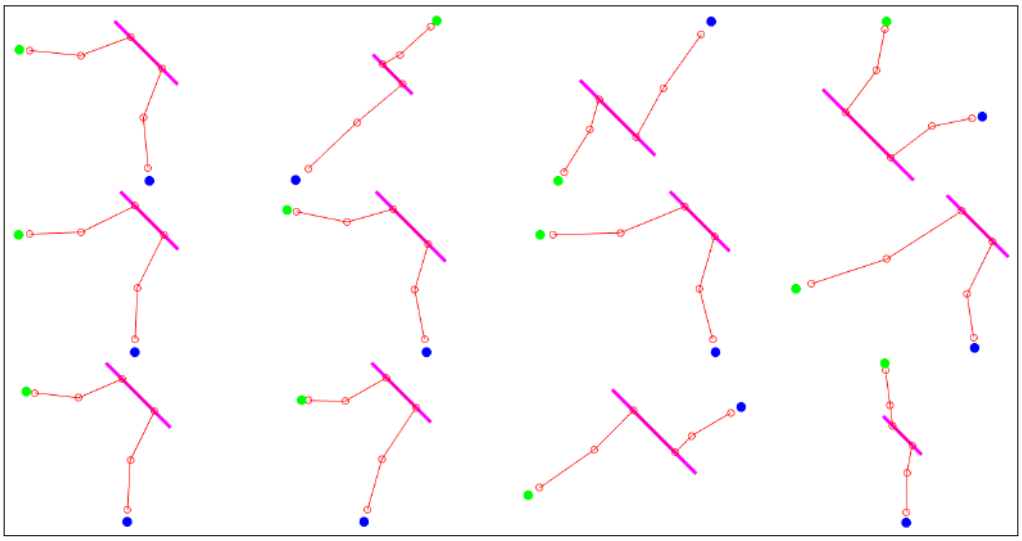


Figure 3. These images show the 2d toy task of the experiment in Section 6.1. The task is to go from a start state (green dot) to a goal state (blue dot). During the motion a contact with the magenta line should be established. The four images on the top row are used as training data and the eight other images are used for testing.

method	error (train)	error (test)	constraint violation (train)	constraint violation (test)
IKKT (feature)	0.027475	0.46944	1.1102e-15	1.6653e-15
IKKT (kernel)	0.94625	66.065	4.4409e-16	8.2469e-16
CIOC	0.014732	0.64592	0.00058039	0.001128

Figure 4. The results from the 2d toy task. The error is the sum of absolute difference between the resulting motion with the learned weights $\boldsymbol{w}$ and the reference motion. The constraint violation is the distance to the magenta line.

such generalization abilities are demonstrated in Section 6 with a box sliding task where we generalized to different box positions and with the door opening task where we generalized to different door angles.

5.5 Nonparametric Imitation Learning

Marinho et al. (2016) represent trajectories as vectors in reproducing kernel Hilbert spaces (RKHS). They propose a functional gradient motion planning algorithm based where trajectories are represented as a linear combination of kernels. The motion planning objective is to minimize a cost functional that maps each trajectory in RKHS to a scalar cost. The cost functional consists of a smoothness term and an obstacle avoidance term. The optimization is done by computing the functional gradient. There approach is similar to our nonparametric variant. Whereas they represent the trajectory in RKHS and define a cost

over these trajectories, we represent the cost function at each time step as a functional in a RKHS. Using functional gradient techniques for imitation learning was proposed by Ratliff et al. (2009), which extends maximum margin planning methods to non-linear cost functions.

Grubb and Bagnell (2010) and Bradley (2009) propose approaches that rely on deep modular systems to learn non-linear cost functions. Functional backpropagation (Grubb and Bagnell 2010) combines functional gradient descent with backpropagation mechanics in Euclidean function space. It allows the use of a greater class of learning algorithms than standard backpropagation. A key aspect of their work is a modular system that separates the structural aspects of the network from the learning in individual modules. Their results show that the functional gradient variant is more robust to local minima than the parameterized gradient.

Levine et al. (2011) use a Gaussian process to learn the reward as a nonlinear function. Additionally to learning the reward they also learn the kernel hyperparameter to recover the structure of the reward function. To do this they maximize the likelihood of the reward under the observed expert demonstrations.

There also has been some research on using Bayesian nonparametric methods for inverse optimal control. Choi and Kim (2013) present for example a Bayesian nonparametric approach to constructing features for the cost function using the Indian buffet process. Michini and How (2012) propose a Bayesian nonparametric inverse reinforcement learning approach that partitions the demonstrations into sets of smaller sub-demonstrations. For each sub-demonstration a simple reward function is learned. The partition process is automated by using a Chinese restaurant process prior as a generative model over partitions. This makes it not necessary to specify the number of partitions by hand. Both formulations are well formulated and very powerful. However, these Bayesian nonparametric inference approaches suffer from the problems of expensive computation and local approximation.

6 Experiments

In the following experimental evaluations we demonstrate the learning properties and the practical applicability of our approach and compare it to alternative methods.

First, we compare our proposed IKKT method on a 2d problem to a state-of-the-art IOC method that does not incorporate constraints. Second, we show on a simple task the ability to reestimate weight functions from optimal demonstrations with different weight parametrizations. Afterwards, we present more complex tasks like sliding a box, opening a door and closing a drawer.

6.1 IOC on a 2d Problem with Constraints

In this evaluation we compare different IOC algorithms on a 2d problem task. We will compare:

- Inverse KKT with a set of features (see Section 3)
- Inverse KKT with a kernel (see Section 4)
- Continuous Inverse Optimal Control (CIOC) that was proposed by Levine and Koltun (2012)

The task is a two dimensional trajectory optimization problem of a point mass. The trajectory consists of 6 time steps that lead to a trajectory $\boldsymbol{x}_{0:T} \in \mathbb{R}^{12}$. The goal of the task is to go from a start state to a goal state. At time step 3 and 4 of the trajectory the robot should be in contact with a line. During this contact phase the robot should move 1 unit in the vertical direction downwards. The state of the environment $\boldsymbol{y}$ contains the initial position, goal position and line parametrization. The domain is visualized in Figure 3.

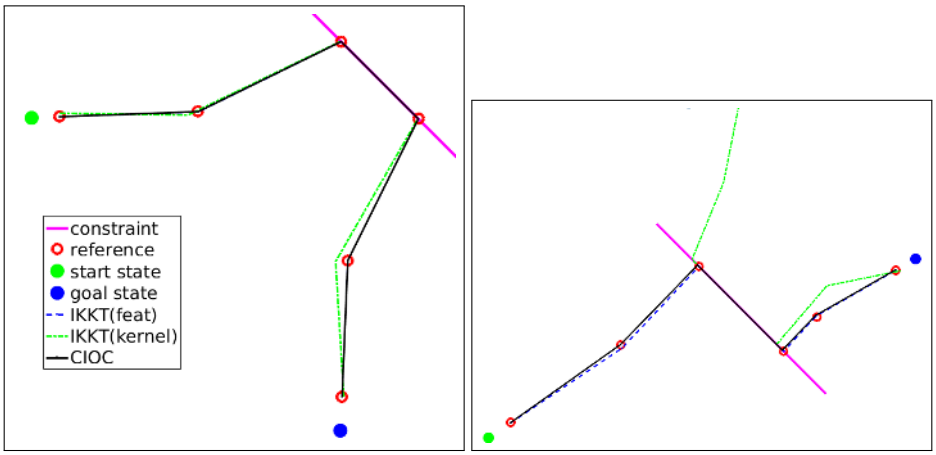


Figure 5. Evaluation of the learned parameter of the 2d point task (Section 6.1). top image shows the performance on a training scenario and the bottom image shows the performance on a test scenario.

We use transition features and a set of linear features around 4 points in the 2d world for both methods. In the two IKKT algorithms we represent the contact in form of equality constraints $\boldsymbol{h}(\boldsymbol{x}_{0:T}, \boldsymbol{y})$. Since the CIOC formulation does not incorporate constraints, we add the contacts directly into the cost features $\boldsymbol{\Phi}(\boldsymbol{x}_{0:T}, \boldsymbol{y})$. Initially, we create 12 motions for different scenarios $\boldsymbol{y}$ (see Figure 3). In the IKKT with kernel variant we augment the state and add the $\boldsymbol{y}$ to the input of the RBF kernel. We split this data in a training and test set. 4 motions are used to train the IOC methods and 8 motions are used for the evaluation.

To evaluate the methods, we first use the training data as input to the IOC methods and learn a weight vector $\boldsymbol{w}$. Afterwards, we use the learned weight vector in the optimal control problem to generate motions for the test scenarios. The resulting motions are compared to the reference motions of the test scenarios. We compare the error of the trajectories and the violation of the constraints on the training and test set. The results are shown in the table in Figure 4. We also visualize the resulting motions of all three variants in Figure 5 for a training scenario (top) and a test scenario (bottom). The results confirm that CIOC and IKKT (feature) reach for the same feature set a similar performance (see discussion in Section 5.1). The nonparametric variant of IKKT achieves a much lower performance. It manages to reach a reasonable training error. However, the generalization abilities are very limited, which is due to the simple RBF kernel at each time step. In order to improve the performance multiple kernel learning methods would be necessary. In the following experiments we will therefore focus on the parametric variant of IKKT. In this evaluation CIOC reached a lower training error and IKKT reached a lower test error. Also the constraint violation of CIOC is higher than for the two IKKT methods since it has to weight the contact features with the other features and IKKT can incorporate them separately as constraints.

6.2 Different Weight Parametrizations in a Benchmark Scenario

The goal of our work is to learn cost functions for finite horizon optimal control problems, including when and how long the costs should be active. In this experiment we test our approach on a simple

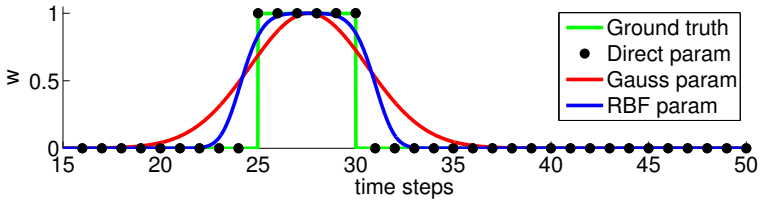


Figure 6. Learned time profiles of different weight parameterizations. For more details see Section 6.2

benchmark scenario. Therefore, we create synthetic demonstrations by optimizing the forward problem with a known ground truth parameter set $\boldsymbol{w}^{\mathrm{GT}}$ and test if it is possible to reestimate these parameters from the demonstrations. We create three demonstrations with 50 time steps, where we define that in the time steps 25 to 30 of these demonstrations the robot end-effector is close to a target position. For this experiments we use a simple robot arm with 7 degree of freedom and the target is a sphere object. We compare the three parametrizations

- **Direct parametrization:** A different parameter is used at each time step (i.e., $\boldsymbol{w} = \boldsymbol{\theta}$) which results in $\boldsymbol{\theta} \in \mathbb{R}^{50}$.
- **Radial basis function:** The basis functions are equally distributed over the time horizon. We use 30 Gaussian basis functions with standard deviation 0.8. This results in $\boldsymbol{\theta} \in \mathbb{R}^{30}$.
- **Nonlinear Gaussian:** A single unnormalized Gaussian weight profile where we have $\boldsymbol{\theta} \in \mathbb{R}^{3}$ with the weight as linear parameter and the nonlinear parameters are the mean and standard deviation. In this case the mean directly corresponds to the time where the activation is highest.

The demonstrations are used as input to our inverse KKT method (see Section 3) and the weights are initialized randomly. A comparison of the learned parameters and the ground truth parameter is shown in Figure 6. The green line represents the ground truth knowledge used for creating the demonstrations. The black dots show the learned parameters of the direct parametrization. The red line shows the learned Gaussian activation and the blue line shows the RBF network. As it can be seen all parametrization detect the right activation region between the time steps 25 to 30 and approximate the ground truth profile. The Gaussian and RBF parametrization also give some weight to the region outside the actual cost region, which is reasonable since in the demonstrations the robot is still close to the target position. After learning with these parametrizations, we conclude that the linear RBF network are best suited to learn time profiles of cost functions. The main reason for this is the linearity of the parametrization that makes the inverse KKT problem convex and the versatility of the RBF network to take on more complex forms. Directly learning the time with the nonlinear Gaussian-shaped parametrization was more difficult and required multiple restarts with different initialization. This demonstrates that the framework of constrained trajectory optimization and its counterpart inverse KKT works quite well for reestimating cost functions of optimal demonstrations.

6.3 IKKT with Noisy Demonstrations

A core assumption of IKKT is that the demonstrations are optimal. In this experiment, we want to investigate what happens if this is not the case. Therefore, we create scenarios with non-optimal

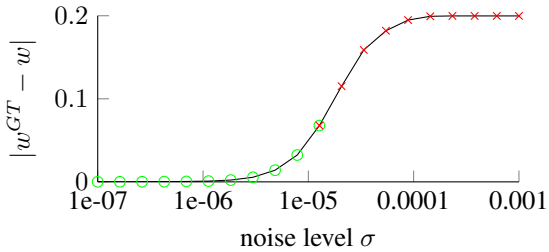


Figure 7. Error in estimating the ground truth parameter for different noise levels.

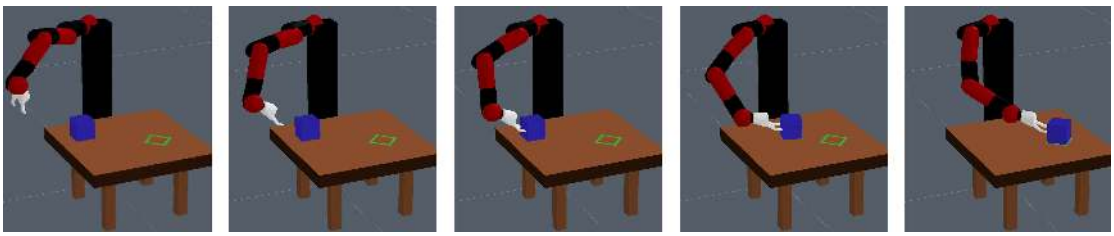


Figure 8. These images show the box sliding motion of Section 6.4 where the goal of the task is to slide the blue box on the table to the green target region.

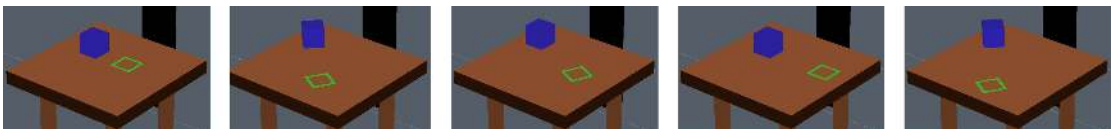


Figure 9. Each image shows a different instance of the box sliding task. We were able to generalize to different initial box states (blue box) and to different final box targets (green area).

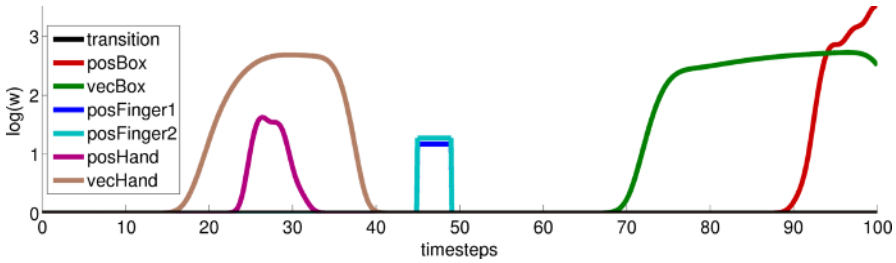


Figure 10. The resulting parameters $\boldsymbol{w}$ of the extracted relevant features plotted over time. task is depicted in this slideshow.

demonstrations and evaluate if IKKT is still able to estimate the underlying cost parameters. We use

Black box IOC:

repeat

Resample parameters $\{\boldsymbol{w}^{(n)}\}_{n=1}^{N}$ with CMA

for all $\boldsymbol{w}^{(n)}$ **do**

Optimize cost function with parameter $\boldsymbol{w}^{(n)}$

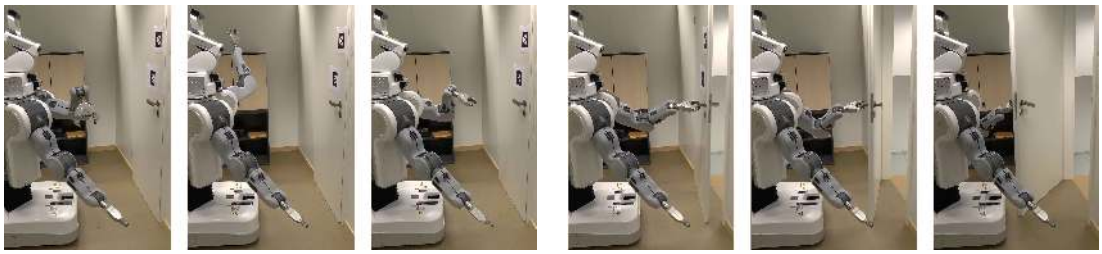
Compute fitness $f^{(n)} = \sum_d (\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}^{(d)})^2$

Update CMA distribution with fitness values

until

Method	$(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}})^2$	**comp. time**
inverse KKT	0.00021	49.29 sec
black box IOC	0.00542	7116.74 sec

Figure 11. On the left side is the black box IOC algorithm we used for comparison in Section 6.4. On the right side are the results of the evaluation that show that our method is superior in terms of squared error between the trajectories and computation time.



(a) (b)

Figure 12. These images show the generalization abilities of our approach. The pictures in (a) show different initial positions of the robot and the pictures in (b) show different final door angle positions. After learning the weight parameter $\boldsymbol{w}^{\star}$ with inverse KKT it was possible to generalize to all these instances of the door opening task.

the same scenario as in the previous experiment where the robot has to reach a target position with the end-effector.

In this scenario we add different levels of Gaussian noise to the optimal demonstrations. We want to test if IKKT is still able to extract the true parameter $\boldsymbol{w}^{\mathrm{GT}}$ that was used to create the noise-free demonstrations. In Figure 7 the absolute error is visualized for different standard deviations σ. The values are averaged over 100 different random seeds. The values where the task could still be performed are visualized with green circles and failues are visualized with red crosses. We defined a successful run when the target was reached inside a 1 cm tolerance. The results show that the estimation error increases continously with the noise level and if σ is above $2\mathrm{e}{-05}$ then the task fails. This demonstrates the requirements to use Inverse KKT only with optimal demonstrations.

6.4 Sliding a Box on a Table

In this experiment we use our approach to learn a cost function for sliding a box on a table. This task is depicted in Figure 8. The goal is to move the blue box on the table to the green marked target position and orientation. The robot consist of a fixed base and a hand with 2 fingers. In total the robot has 10 degrees of freedom. Additionally to these degree of freedom we model the box as part of the configuration state, which adds 3 more degrees of freedom (2 translational + 1 rotational). The final box position and orientation is provided as input to our approach and part of the external parameters $\boldsymbol{y}$. We used three synthetic demonstrations of the task and created a set of features with the approach described in Section 3.4 that led to $\theta \in \mathbb{R}^{537}$ parameters. The relevant features extracted from our algorithm are

- **transition:** Squared acceleration at each time step in joint space
- **posBox:** Relative position between the box and the target.
- **vecBox:** Relative orientation between the box and the target.
- **posFinger1/2:** Relative position between the robots fingertips and the box.
- **posHand:** Relative position between robot hand and box.
- **vecHand:** Relative orientation between robot hand and box.

The contacts between the fingers and the box during the sliding are modeled with equality constraints. They ensure that during the sliding the contact is maintained. For achieving realistic motions, we use an inequality constraint that restrict the movement direction during contact into the direction in which the contact is applied. This ensures that no unrealistic motions like sliding backwards or sidewards are created. For clarity we would like to note that we are not doing a physical simulation of the sliding behavior in these experiments. Our goal was more to learn a policy that executes a geometric realistic trajectory from an initial to a final box position. Figure 8 shows one of the resulting motion after learning. We were able to generalize to a wide range of different start and goal position of the box (see Figure 9). Videos of the resulting motions can be found in the supplementary material.

We compare our method to a black-box optimization approach similar to (Mombaur et al. (2010); Rückert et al. (2013)). We implemented this approach with the black-box method Covariance Matrix Adaptation (CMA) by Hansen and Ostermeier (2001) in the outer loop and our constrained trajectory optimization method (see Section 2) in the inner loop. The resulting algorithm is described in Figure 11. As fitness function for CMA we used the squared distance between the current solution $\boldsymbol{x}^{(n)}$ and the demonstrations $\hat{\boldsymbol{x}}^{(d)}$. We compare this method with our inverse KKT approach by computing the error between the solution and demonstrations and the computational time, which are shown in the table in Figure 11. The black-box method took around 4900 iterations of the outer loop of the above algorithm until it converged to a solution. This comparison shows that using structure and optimality conditions of the solution can enormously improve the learning speed. Further difficulties with black box methods like CMA is that they cannot naturally deal with constraints (in our case $\boldsymbol{w} > \mathbf{0}$) and that the initialization is non-trivial.

6.5 Opening a Door with a PR2

In this experiment we apply the introduced inverse KKT approach from Section 3 on a skill with the goal to open a door with a real PR2 robot. The problem setup is visualized in Figure 1. We use a model of the door for our planning approach and track the door angle with AR marker. We use the left arm of the

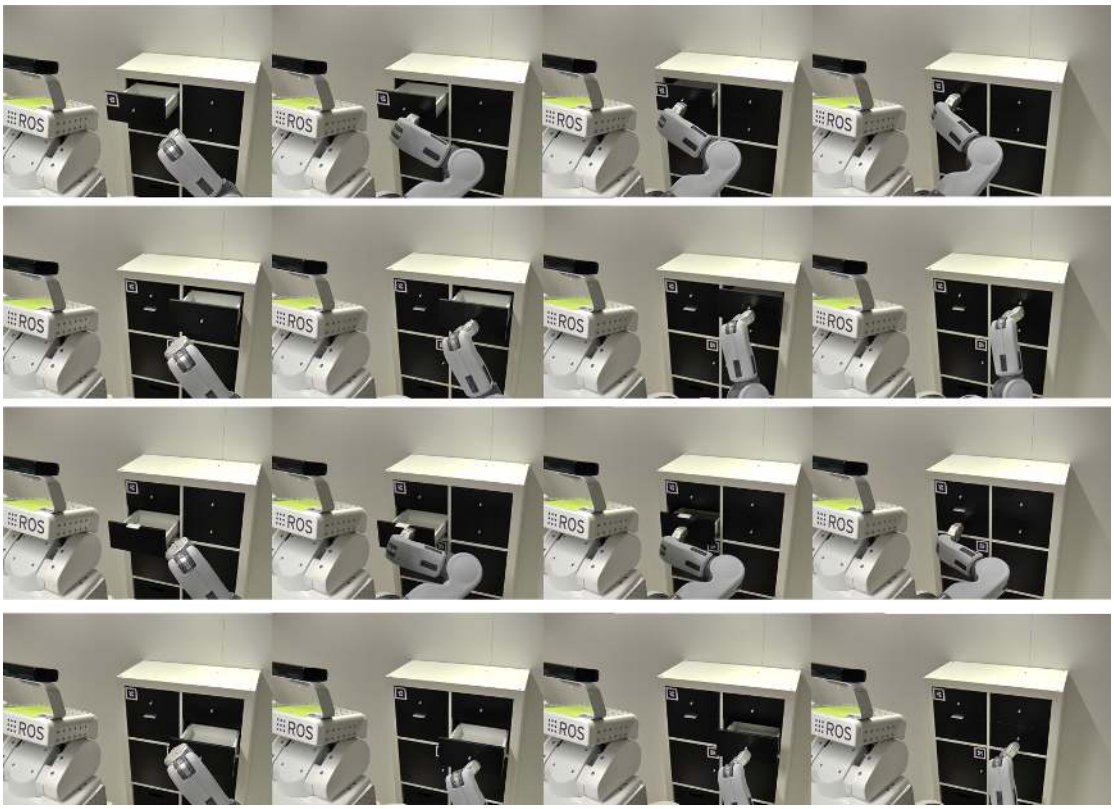


Figure 13. Resulting motions of the drawer closing experiments with the PR2 (see Section 6.6). We used two demonstrations for different drawers as input for our inverse KKT method. Each row shows a resulting motion after optimizing the cost function for a different drawer of the shelf.

robot that consists of 7 rotational joints and also include the door angle as configuration state into $\boldsymbol{x}$. This allows us to define cost functions directly on the door angle. The gripper joint is fixed during the whole motion. For our IOC algorithm we recorded 2 demonstrations of opening the door from different initial positions with kinesthetic teaching. The motions also include the unlocking of the door by turning the handle first. During the demonstrations we also recorded the door position with the attached markers. We created a feature set similar to the box sliding motion from the previous experiment. Our inverse KKT algorithm extracted the features:

- Relative position & orientation between gripper and handle before and after unlocking the handle.
- end-effector orientation during the whole opening motion.
- Position of the final door state.

We use equality constraints, similar to the box sliding experiment to keep the contact between end-effector and door. Furthermore, we use inequality constraints to avoid contacts with the rest of the robot

body. We are able to robustly generate motions with these parameters that generalize to different initial positions and different target door angles (see Figure 12). Videos of all these motions can be found in the supplementary material.

6.6 Closing Drawers with PR2

In this experiment we applied the introduced IOC approach from Section 3 on a skill with the goal to close a drawer with a real PR2 robot. The problem setup is visualized in Figure 13. The shelf we focus on in this experiments has four drawers at different positions. The drawer position was part of the external parameters $\boldsymbol{y}$ that allows us to adapt the motion to different drawers. We used the right arm of the robot that consists of 7 rotational joints. As demonstrations we provided 2 trajectories for 2 different drawers by kinesthetic teaching. During the demonstrations we also recorded the positions of the drawer by using AR markers. For our IOC algorithm we provided 9 different features and 2 constraints, which are similar to the ones of the door task. We were able to generate motions with these parameters that generalized to all four drawers. The resulting motions are visualized in Figure 13.

7 Conclusion

In this paper we introduced inverse KKT motion optimization, an inverse optimal control method for learning cost functions for constrained motion optimization problems. Our formulation is focused on finite horizon optimal control problems for tasks that include contact with the environment. The resulting method is based on the KKT conditions that the demonstrations should fulfill. We proposed a formulation that uses a weighted sum of squared features as cost function and a formulation that represents the cost function as a functional in a reproducing kernel Hilbert space. For a typical linear parameterization of cost functions this leads to a convex problem; in the general case it is implemented as a 2nd order optimization problem, which leads to a fast convergence rate. We demonstrated the method in a real robot experiment of opening a door that involved contact with the environment. In our future research we plan to further automate and simplify the skill acquisition process. Thereby, one goal is to extend the proposed method to be able to handle demonstrations that are not recorded on the robot body. Another goal is to couple it with reinforcement learning methods to improve the performance over the demonstrated behavior.

Acknowledgements

This work was supported by the DFG under grant TO 409/9-1.

References

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

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM.

Albrecht S, Ramirez-Amaro K, Ruiz-Ugalde F, Weikersdorfer D, Leibold M, Ulbrich M and Beetz M (2011) Imitating human reaching motions using physically inspired optimization principles. In: *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*.

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

Boularias A, Kober J and Peters J (2011) Relative Entropy Inverse Reinforcement Learning. In: *Proceedings of Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*. URL 120.

Bradley D (2009) *Learning In Modular Systems*. PhD Thesis, Carnegie Mellon University.

Calinon S, Alizadeh T and Caldwell DG (2013) On improving the extrapolation capability of task-parameterized movement models. In: *International Conference on Intelligent Robots and Systems*.

Choi J and Kim KE (2013) Bayesian nonparametric feature construction for inverse reinforcement learning. In: *Proceedings of the international joint conference on Artificial Intelligence*.

Doerr A, Ratliff N, Bohg J, Toussaint M and Schaal S (2015) Direct Loss Minimization Inverse Optimal Control. In: *Proceedings of Robotics: Science and Systems*.

Englert P and Toussaint M (2015) Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations. In: *Proceedings of the International Symposium of Robotics Research*.

Finn C, Levine S and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: *Proceedings of the International Conference on Machine Learning*.

Gönen M and Alpaydın E (2011) Multiple kernel learning algorithms. *The Journal of Machine Learning Research* 12: 2211–2268.

Grubb A and Bagnell JA (2010) Boosted backpropagation learning for training deep modular networks. In: *nternational Conference on Machine Learning*.

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

Jetchev N and Toussaint M (2014) TRIC: Task space retrieval using inverse optimal control. *Autonomous Robots* 37(2): 169–189.

Kalakrishnan M, Pastor P, Righetti L and Schaal S (2013) Learning objective functions for manipulation. In: *Proceedings of the International Conference on Robotics and Automation*.

Kolter JZ, Abbeel P and Ng AY (2008) Hierarchical apprenticeship learning with application to quadruped locomotion. In: *Neural Information Processing Systems*.

Levine S and Koltun V (2012) Continuous inverse Optimal Control with Locally Optimal Examples. In: *Proceedings of the International Conference on Machine Learning*.

Levine S, Popovic Z and Koltun V (2011) Nonlinear inverse reinforcement learning with gaussian processes. In: *Neural Information Processing Systems*.

Marinho Z, Boots B, Dragan A, Byravan A, Gordon GJ and Srinivasa S (2016) Functional gradient motion planning in reproducing kernel hilbert spaces. In: *Proceedings of Robotics: Science and Systems*.

Michini B and How JP (2012) Bayesian nonparametric inverse reinforcement learning. In: *European Conference on Machine Learning*.

Mombaur K, Truong A and Laumond JP (2010) From human to humanoid locomotionan inverse optimal control approach. *Autonomous Robots* 28(3): 369–383.

Muhlig M, Gienger M, Steil JJ and Goerick C (2009) Automatic selection of task spaces for imitation learning. In: *Proceedings of the International Conference on Intelligent Robots and Systems*.

Ng AY and Russell S (2000) Algorithms for Inverse Reinforcement Learning. In: *Proceedings of the International Conference on Machine Learning*.

Nocedal J and Wright SJ (2006) *Numerical Optimization*, volume 2. Springer.

Paraschos A, Daniel C, Peters J and Neumann G (2013) Probabilistic Movement Primitives. In: *Neural Information Processing Systems*.

Pastor P, Kalakrishnan M, Chitta S, Theodorou E and Schaal S (2011) Skill learning and task outcome prediction for manipulation. In: *Proceedings of the International Conference on Robotics and Automation*.

Puydupin-Jamin AS, Johnson M and Bretl T (2012) A convex approach to inverse optimal control and its application to modeling human locomotion. In: *Proceedings of the International Conference on Robotics and Automation*.

Ramachandran D and Amir E (2007) Bayesian inverse reinforcement learning. In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. pp. 2586–2591.

Ratliff ND, Bagnell JA and Zinkevich M (2006) Maximum margin planning. In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. pp. 729–736.

Ratliff ND, Silver D and Bagnell JA (2009) Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots* 27: 25–53. DOI:10.1007/s10514-009-9121-3.

Rückert EA, Neumann G, Toussaint M and Maass W (2013) Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience* 6.

Schaal S, Ijspeert A and Billard A (2003) Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions of the Royal Society of London* 358: 537–547.

Schölkopf B and Smola AJ (2002) *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Stulp F, Raiola G, Hoarau A, Ivaldi S and Sigaud O (2013) Learning compact parameterized skills with a single regression. In: *Proceedings of the International Conference on Humanoid Robots*.

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: *Geometric and Numerical Foundations of Movements*. Springer.

Toussaint M, Ratliff N, Bohg J, Righetti L, Englert P and Schaal S (2014) Dual Execution of Optimized Contact Interaction Trajectories. In: *Proceedings of the IEEE International Conference on Robotics and Automation*.

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 and Dey AK (2008) Maximum Entropy Inverse Reinforcement Learning. *In Proceedings of the AAAI Conference on Artificial Intelligence* .

A Derivation of IKKT Loss

The loss of a single demonstration is according to Equation (12) given by

$$\begin{aligned}
\ell^{(d)}(\boldsymbol{w}, \boldsymbol{\lambda}^{(d)}) &= \left|\left|\nabla_{\boldsymbol{x}_{1:T}} L(\hat{\boldsymbol{x}}_{0:T}^{(d)}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\lambda}^{(d)}, \boldsymbol{w})\right|\right|^2 \\
&= (2\boldsymbol{J}^\top \text{diag}(\boldsymbol{w})\Phi + \boldsymbol{J}_c^\top \boldsymbol{\lambda}^{(d)})^\top (2\boldsymbol{J}^\top \text{diag}(\boldsymbol{w})\Phi + \boldsymbol{J}_c^\top \boldsymbol{\lambda}^{(d)}) \\
&= 4\boldsymbol{\Phi}^\top \text{diag}(\boldsymbol{w}) \boldsymbol{J}\boldsymbol{J}^\top \text{diag}(\boldsymbol{w})\boldsymbol{\Phi} + \boldsymbol{\lambda}^{(d)} \boldsymbol{J}_c \boldsymbol{J}_c^\top \boldsymbol{\lambda}^{(d)} \\
&\quad + 2\boldsymbol{\Phi}^\top \text{diag}(\boldsymbol{w}) \boldsymbol{J}\boldsymbol{J}_c^\top \boldsymbol{\lambda}^{(d)} + 2\boldsymbol{\lambda}^{(d)\top} \boldsymbol{J}_c \boldsymbol{J}^\top \text{diag}(\boldsymbol{w})\boldsymbol{\Phi} \, .
\end{aligned}$$

Inserting $\boldsymbol{\lambda}^{(d)}$ from Equation (14) into this loss function and rearranging the terms results in the IOC cost per demonstration given in Equation (15):

$$\begin{aligned}
\ell^{(d)}(\boldsymbol{w}) = 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} + 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \\
\boldsymbol{J}\tilde{\boldsymbol{J}_c}^\top (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)(\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} \tilde{\boldsymbol{J}_c}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} \\
- 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J}\tilde{\boldsymbol{J}_c}^\top (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} \tilde{\boldsymbol{J}_c}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} \\
- 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J}\tilde{\boldsymbol{J}_c}^\top (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} \tilde{\boldsymbol{J}_c}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} \\
= 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} \\
- 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J}\tilde{\boldsymbol{J}_c}^\top (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} \tilde{\boldsymbol{J}_c}\boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w} \\
= 4\boldsymbol{w}^\top \text{diag}(\boldsymbol{\Phi}) \boldsymbol{J} \left(\boldsymbol{I} - \tilde{\boldsymbol{J}_c}^\top (\tilde{\boldsymbol{J}_c}\tilde{\boldsymbol{J}_c}^\top)^{-1} \tilde{\boldsymbol{J}_c} \right) \boldsymbol{J}^\top \text{diag}(\boldsymbol{\Phi})\boldsymbol{w}.
\end{aligned}$$