Inverse KKT Motion Optimization: A Newton Method to Efficiently Extract Task Spaces and Cost Parameters from Demonstrations

Peter Englert
Machine Learning and Robotics Lab
Universität Stuttgart
Germany

Marc Toussaint
Machine Learning and Robotics Lab
Universität Stuttgart
Germany

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. Inverse KKT motion optimization assumes that the demonstrations fulfill the Karush-Kuhn-Tucker conditions of an unknown underlying constrained optimization problem, and extracts parameters of this underlying problem. We propose the latter to extract the relevant task spaces and cost parameters from demonstrations of skills that involve contacts. The aim of this 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 introduce the novel approach and present first results on simulation and real robots.

1 Introduction

The following work is in the context of skill acquisition for robots through imitation. Cost functions are a widely-used 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 abilities for a wide range of problem configurations. However, designing powerful 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, has been introduced [Russell, 1998], which 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.

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. 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.

Our approach is based on finding a cost function, such that the demonstrations fulfill the KKT conditions of an underlying constrained optimization problem with this cost function. The main assumption is that other solutions, which also fulfill the KKT conditions of the constrained optimization problem, will be similar to the demonstrations. Our method learns cost functions, including the identification of relevant task spaces, for constrained motion optimization. We formulate the IOC problem as a 2nd order optimization problem that we solve with Newton's method. We integrate constraints into the IOC method which allows us to incorporate manipulations of objects within the motion.

The structure of the remaining paper is the following. First, we present in Section 2 a short overview on related work on inverse optimal control. Then in Section 3, we introduce some background on constrained trajectory optimization. Afterwards, we develop our IOC algorithm in Section 4. In the experimental section 5 we evaluate our approach on simulated and real robot experiments.

The main contribution of this paper is the introduction of an IOC method for constrained motion optimization that is based on the KKT conditions. Our method is formulated as a 2nd order optimization problem and therefore allows to efficiently extract task spaces and parameters from demonstrations.

2 Related Work

In the recent years there has been extensive research on inverse optimal control. In the following we will focus on the approaches and methods that are most related to our work. For a good overview on inverse optimal control approaches we refer the reader to the survey paper of [Zhifei and Joo, 2012].

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 of [Ziebart et al., 2008]. The learning procedure is performed by maximizing the 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 Newton method. Further, our formulation also targets at efficiently extracting the relevant task spaces.

Another approach are black box approaches to find cost parameters for motions [Mombaur et al., 2010, Rückert et al., 2013]. There, usually a two-layered optimization procedure is used, where in the outer loop black box optimization methods (e.g., covariance matrix adaptation) are used to find suitable parameter of the inner motion problem. Such methods usually have high computational costs for higher-dimensional spaces since the black box optimizer needs many evaluations.

In [Jetchev and Toussaint, 2011] they discover task relevant features by training a 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. However, our approach more rigorously extracts task dimensions in the inverse KKT motion optimization framework, including motions that involve contacts.

3 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 $\boldsymbol{x}^\star_{1:T}$, given an initial configuration $\boldsymbol{x}_0$, that minimizes a certain objective function

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

We define trajectory optimization in Equation (1) as a sum over k-order cost functions, where each cost term depends on k-order states $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_{t-k}, \dots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ that contain the current and k previous robot configurations [Toussaint, 2014]. This also allows 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 differ between different problem instances, which is used to generalize the skill to different environment configurations. The cost terms in Equation (1) are given as

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

where $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ are the features and $\boldsymbol{w}_t$ is the weighting vector at time t. This means the cost at one time-step is defined as the weighted sum of squared features.

A simple example for one feature type is that the endeffector of the robot at the end of the motion T should be at the position of a cup. In this example the feature $\boldsymbol{\phi}_T(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ would compute the difference between the forward kinematics mapping and cup position (given by $\boldsymbol{y}$). More complex tasks define body orientations or relative positions between robot and an object. Transitions 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 constraints

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

which are also defined with features. An example for inequality constraints 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 difference between the distance of the robot body to the obstacle and the allowed threshold.

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

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

and the overall optimization problem as

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

We take the inequalities with the Augmented Lagrangian method [Nocedal and Wright, 2006] into account. Therefore, additionally to the solution $\boldsymbol{x}^\star_{1:T}$ we also get the Lagrange parameters $\boldsymbol{\lambda}^\star_{1:T}$, which provide information 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 trajectory optimization problem. Therefore we compute the Jacobian of the features $\boldsymbol{J} = \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{x}}$ and the Jacobian of the inequality constraints $\boldsymbol{J}_g = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$. The gradient is

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

and the Hessian is approximated as in Gauss-Newton as

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

With this framework we are able to produce motions that minimize an objective function that consist of a weighted combination of different features. Additionally, we are also able to include constraints of the environment and obtain information when the constraints are active.

4 Inverse KKT Motion Optimization

In the following section, we introduce an approach to learn the weight vector $\boldsymbol{w}$ in Equation (5) from demonstrations. 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 $(\hat{\boldsymbol{x}}^{(d)}_{1:T}, \hat{\boldsymbol{y}}^{(d)})^D_{d=1}$. We introduce a linear parametrization $\boldsymbol{w}(\boldsymbol{\theta}) = A\boldsymbol{\theta}$ to reduce the amount of parameters, where $\boldsymbol{\theta}$ are the parameters that the IOC method learns. This parametrization allows a flexible assignment of one parameter to multiple task costs. For example the same weight parameter could be constant over all time steps in a collision avoidance task

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

$$L(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, A\boldsymbol{\theta}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \tag{8}$$

and the first Karush-Kuhn-Tucker condition, which says that for an optimal solution $\boldsymbol{x}^\star_{1:T}$ the condition $\nabla_{\boldsymbol{x}_{1:T}} L(\boldsymbol{x}^\star_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \boldsymbol{0}$ has to be fulfilled. We assume that the demonstrations are optimal and should fulfill this conditions. Therefore, the IOC problem can be viewed as searching for parameter $\boldsymbol{\theta}$, such that this condition is fulfilled for all the demonstrations.

We formulate this idea into the loss function

$$J(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \sum_{d=1}^{D} J^{(d)}(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(d)}) \tag{9}$$

$$J^{(d)}(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(d)}) = \Big(\nabla_x L(\hat{\boldsymbol{x}}_{1:T}^{(d)}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\lambda}^{(d)}, \boldsymbol{\theta})\Big)^\top \Big(\nabla_x L(\hat{\boldsymbol{x}}_{1:T}^{(d)}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\lambda}^{(d)}, \boldsymbol{\theta})\Big) \ , \tag{10}$$

where d enumerates the demonstrations and $\boldsymbol{\lambda}^{(d)}$ is the dual to the demonstration $\hat{\boldsymbol{x}}_{1:T}^{(d)}$ under the problem defined by $\boldsymbol{\theta}$. Therefore $\boldsymbol{\lambda}^{(d)} = \boldsymbol{\lambda}^{(d)}(\hat{\boldsymbol{x}}_{1:T}^{(d)}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\theta})$ is a function of the primal demonstration, the environment configuration of that demonstration, and the underlying parameters $\boldsymbol{\theta}$. Further, $J(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(d)}(\boldsymbol{\theta})) = J(\boldsymbol{\theta})$ becomes a function of the parameters only.

We minimize $J(\boldsymbol{\theta})$ with Newton's method [Nocedal and Wright, 2006]. In each Newton iteration we compute $\boldsymbol{\lambda}^{(d)}(\boldsymbol{\theta})$ for each demonstration separately by choosing the dual solution that again minimizes the error in the first KKT condition for the respective demonstration,

$$\frac{\partial}{\partial \boldsymbol{\lambda}^{(d)}} J^{(d)}(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(d)}) = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\lambda}^{(d)}(\boldsymbol{\theta}) = -(\boldsymbol{J}_g \boldsymbol{J}_g^\top)^{-1} \boldsymbol{J}_g \nabla_x \boldsymbol{f}. \tag{11}$$

This is subject to the KKT's complementarity condition, we only solve for the $\boldsymbol{\lambda}$ that are non-zero for constraints that were active during the demonstration, which we can evaluate as $\boldsymbol{g}$ is independent of $\boldsymbol{\theta}$. By inserting Equation (11) into Equation (9) we get

$$J^{(d)}(\boldsymbol{\theta}) = \nabla_x \boldsymbol{f}^\top \underbrace{\Big(\boldsymbol{I} - \boldsymbol{J}_g^\top (\boldsymbol{J}_g \boldsymbol{J}_g^\top)^{-1} \boldsymbol{J}_g\Big)}_{\boldsymbol{\Lambda}} \nabla_x \boldsymbol{f} = 4 \boldsymbol{\Phi}^\top \mathrm{diag}(\boldsymbol{A}\boldsymbol{\theta}) \boldsymbol{J} \boldsymbol{\Lambda} \boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{A}\boldsymbol{\theta}) \boldsymbol{\Phi} \ . \tag{12}$$

The resulting optimization problem is

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{s.t.} \quad \boldsymbol{\theta} \geq \mathbf{0} \tag{13}$$

Note that we constrain the parameters $\boldsymbol{\theta}$ to be positive. This reflects that we want squared cost features to only positively contribute to the overall cost (4).

Further, the above formulation may lead to the singular solution $\boldsymbol{\theta} = \mathbf{0}$ 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 (4) at the demonstrations (cf. [Levine and Koltun, 2012]). Second, as the absolute scaling of (4) is arbitrary we may additionally add the constraint

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{s.t.} \quad \boldsymbol{\theta} \geq \mathbf{0} \ , \quad \|\boldsymbol{\theta}\|^2 \geq 1 \tag{14}$$

to our problem formulation (13). We choose the latter option in our experiments.

For completeness, the gradient and Hessian for the Newton method are

$$\tfrac{\partial}{\partial \boldsymbol{\theta}} J^{(d)}(\boldsymbol{\theta}) = 8 \Big(\mathrm{diag}(\boldsymbol{\Phi}) \boldsymbol{J} \boldsymbol{\Lambda} \boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{A}\boldsymbol{\theta}) \boldsymbol{\Phi}\Big) \boldsymbol{A} \tag{15}$$

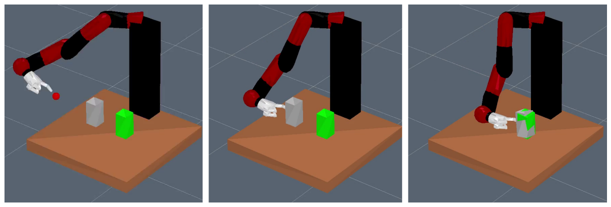
$$\tfrac{\partial^2}{\partial \boldsymbol{\theta}^2} J^{(d)}(\boldsymbol{\theta}) = 8 \boldsymbol{A}^\top \Big(\mathrm{diag}(\boldsymbol{\Phi}) \boldsymbol{J} \boldsymbol{\Lambda} \boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{\Phi})\Big) \boldsymbol{A} \tag{16}$$

5 Experiments

In the following section we evaluate the introduced method of Section 4 on two experiments. In the first experiment we show in a synthetic box moving task in simulation that it is possible to reestimate ground truth parameters with our method. In the second experiment we demonstrate the real world applicability with real demonstrations and a PR2 robot.

5.1 Sliding a Box in Simulation

In this example the task is to slide a box on a table with a robot arm (see Figure 1(a)). We want to evaluate if our method is able to reestimate the task parameters $\boldsymbol{\theta}$ from demonstrations. Therefore,



(a) Robot sliding a box on a table.

Feature	Ground Truth θ	Learned θ
Transition	0.0223038	0.0223377
pos C	0.705309	0.706562
pos T	7.05309	7.1207
rot C	0.00705309	0.00704786
rot T	7.05309	6.98469
vel C	0.0705309	0.0706365
contact	0.0705309	0.0704956

(b) Reestimation of original parameters.

Figure 1: The task of the robot is to slide the grey box to the target location (green square) on the table (see Section 5.1). The Table in (b) shows a comparison between the ground truth parameter with the learned parameter $\boldsymbol{\theta}$.

we first define a ground truth parameter set $\boldsymbol{\theta}$ and generate motions with the constrained trajectory optimization method (see Section 3). Afterwards we use these demonstrations as input to our inverse KKT motion optimization method. The learned parameters are shown in Table 1(b) split into different feature types. The type of features we included were the endeffector position (pos), endeffector rotation (rot), and endeffector velocity (vel). We used these features at two points in time during the motion, which were the final time (F) and the time where the robot gets into contact with the box (C). Additionally we used a feature that ensures to keep the contact between the box and the table (contact), which is active from the time of contact C to the end. By comparing the values between ground truth and learned $\boldsymbol{\theta}$ it can be seen that our method is able to reestimate the ground truth parameters up to some small difference.

5.2 Closing Drawers with PR2

In this experiment we applied the introduced IOC approach from Section 4 on a skill with the goal to close a drawer with a real PR2 robot. The problem setup is visualized in Figure 6. 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[1]. For our IOC algorithm we provided 9 different features (see Table 2), which have similar types to the ones in the previous section. We used two contact features to ensure that the motion is performed towards the degree of freedom of the drawer. The learned weight parameter $\boldsymbol{\theta}$ are shown in Figure 2 We were able to generate motions with these parameters that generalized to all four drawers. The resulting motions are visualized in Figure 6.

Feature	Learned θ
Transition	0.1144
pos C	0.4890
pos T	0.9206
rot C	0.0785
rot T	9.9446
vel C	0.0008
vel T	0.0057
contact 1	0.0047
contact 2	0.0010

Figure 2: Learned parameter.

6 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 method is based on the KKT conditions that the demonstrations should fulfill. 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 that involved contact with the environment. In our future research we plan to further automate the skill acquisition process. Thereby, one goal is the automatic extraction of interesting key points from demonstrations that leads towards an automatic feature generation process. Also the use of nonlinear weight functions $\boldsymbol{w}(\boldsymbol{\theta})$ seems to be interesting to investigate. We also want to connect our approach to symbolic planning methods that allows the sequencing of multiple skills to reach higher level goals.

[1]`http://wiki.ros.org/ar_track_alvar`

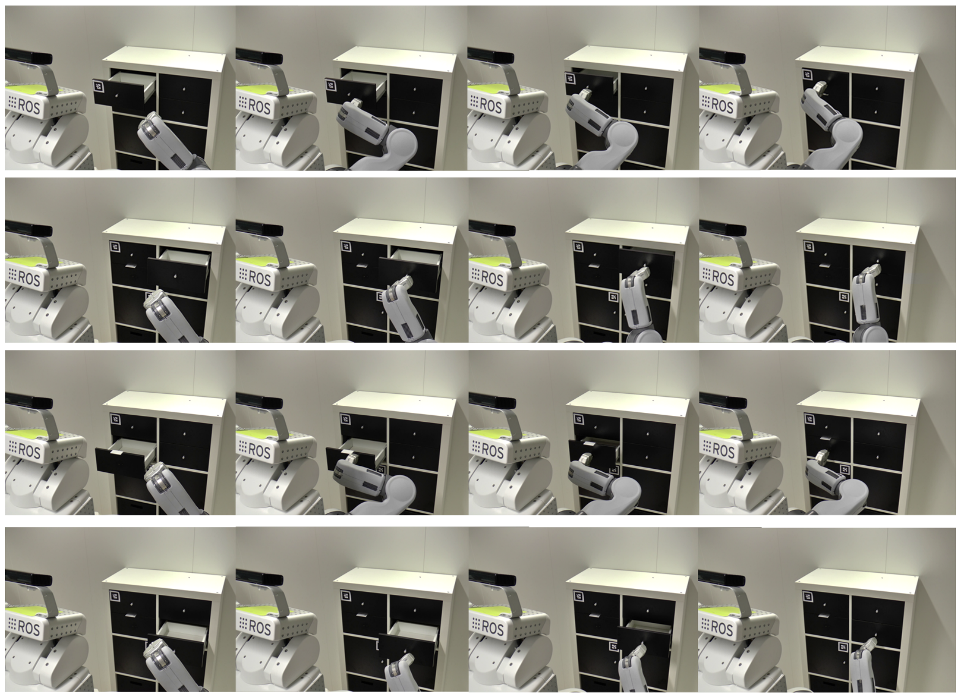


Figure 3: Resulting motions of the drawer closing experiments with the PR2 (see Section 5.2). The figure shows the motions of the PR2 for the four different drawers of the shelf.

References

[Jetchev and Toussaint, 2011] Jetchev, N. and Toussaint, M. (2011). Task Space Retrieval Using Inverse Feedback Control. *Proceedings of the International Conference on Machine Learning*.

[Levine and Koltun, 2012] Levine, S. and Koltun, V. (2012). Continuous Inverse Optimal Control with Locally Optimal Examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*.

[Mombaur et al., 2010] Mombaur, K., Truong, A., and Laumond, J.-P. (2010). From Human to Humanoid Locomotion–An Inverse Optimal Control Approach. *Autonomous Robots*, 28(3):369–383.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). Numerical Optimization 2nd.

[Rückert et al., 2013] Rückert, E. A., 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*.

[Russell, 1998] Russell, S. (1998). Learning Agents for Uncertain Environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103. ACM.

[Toussaint, 2014] Toussaint, M. (2014). Newton methods for k-order Markov Constrained Motion Problems. *arXiv:1407.0414 [cs.RO].*

[Toussaint et al., 2014] 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 Intelligent Robotics Systems*.

[Zhifei and Joo, 2012] Zhifei, S. and Joo, E. M. (2012). A Survey of Inverse Reinforcement Learning Techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311.

[Ziebart et al., 2008] Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. *In Proceedings of the AAAI*.