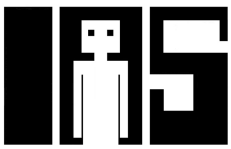
Model-based Imitation Learning by Probabilistic Trajectory Matching

Master-Thesis von Peter Englert
Februar 2013

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Model-based Imitation Learning by Probabilistic Trajectory Matching

Vorgelegte Master-Thesis von Peter Englert

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr.-Ing. Marc Peter Deisenroth

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 26. Februar 2013

(P. Englert)

Abstract

Efficient skill acquisition is crucial for creating versatile robots. One intuitive way to teach a robot new tricks is to enable it to match its behavior to a teacher's demonstration of the task at hand. This approach is known as imitation learning. Classical methods of imitation learning suffer substantially when the actions (i.e., motor commands, torques or forces) of the teacher are not observed and the body of the teacher differs substantially, e.g., in the actuation (which is known as the correspondence problem). Addressing these drawbacks, we propose to train a robot-specific controller that directly matches robot trajectories with observed ones. We present a novel and robust probabilistic model-based approach for solving the trajectory matching problem via policy search. We learn a probabilistic model of the system, which allows mental rehearsal of the current controller by making predictions about future state distributions. These internal simulations allow us to improve the current controller without continuously interacting with the real system, which results in a reduced interaction time. Using long-term predictions from this learned model, we train robot-specific controllers that reproduce the expert's distribution of demonstrations without having observed his motor commands. We demonstrate that our method reaches a higher learning speed than trial-and-error based learning systems with manually generated reward functions. The power of the resulting approach is shown by imitating human behavior using a tendon-driven, compliant robotic arm with complex dynamics.

Contents

1 Introduction

Programming robots to perform complex tasks is difficult with classical methods for instructing robots such as textual or GUI-driven programming techniques [Biggs and Macdonald, 2003]. These methods require a large amount of work for programming a single task and transfer to new environments may often be difficult. Especially for programming versatile robots, where fast learning of new tasks in changing environments is necessary, these methods are often impracticable.

Imitation learning (IL) is an approach to solve such skill acquisition problems in an elegant way: The teacher's demonstration of a task is recorded and subsequently learning algorithms transfer the task to a robot [Argall et al., 2009, Atkeson and Schaal, 1997]. Especially for tasks that humans can perform well, this approach is often easier and more comfortable for transferring skills than classical programming methods. Another advantage is that if robot movements resemble human movements, they will be accepted more easily by humans, which, from a psychological point of view, is desirable when integrating robots into our environment (e.g., for domestic robots). Research in the field of IL devised techniques which differ in the way the demonstrations are provided (e.g., motion capture [Ude et al., 2004], physical interaction [Ben Amor et al., 2009]), the level at which the imitation happens (e.g., at the symbolic [Zöllner et al., 2004] or trajectory level [Calinon et al., 2007]), whether they use a system model, and whether/how they employ reward functions for the task.

The transfer of a skill through imitation learning limits the performance of the robot to the skill of the teacher that provided the demonstration. Reinforcement Learning (RL) is a common technique to improve skills after applying imitation learning. RL [Sutton and Barto, 1998] is an approach, where a task-specific reward function is maximized. RL has been successfully used in robotics applications for learning the ball-in-a-cup game [Kober and Peters, 2010] and flying a helicopter [Ng et al., 2003]. However, a major difficulty in RL is engineering of a suitable reward function for more complex tasks.

Inverse Reinforcement Learning (IRL) is a form of imitation learning that addresses the problem of automatically extracting a reward function from demonstrations of a task [Ng and Russell, 2000, Boularias et al., 2011]. Hence, it is suited for tasks where the hand-crafted definition of a suitable reward function is difficult (e.g., Parking Lot Navigation [Abbeel et al., 2008]). Drawbacks of IRL are that the performance of most methods rely on feature selection which can strongly bias the performance.

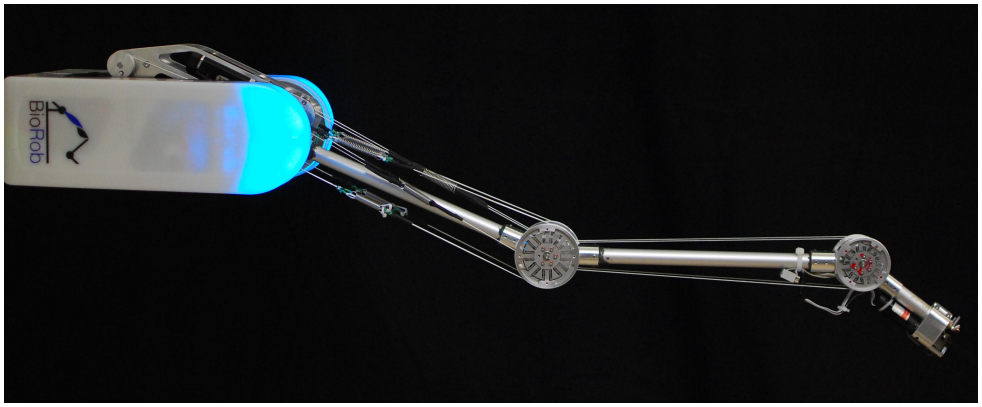


Figure 1.1: The BioRob™ is a compliant, biomechanically-inspired robot manipulator with drive cables and springs, which represent tendons and their elasticity. We evaluate our imitation learning approach on this system by imitating human demonstrations. Classical control approaches based on rigid body dynamics are unrealistic for this robot because they omit the cable-driven properties and the elasticity of the tendons. Therefore, we learn a forward model of the robot's dynamics, which we use for internal simulations to learn a policy.

Another classic form of imitation learning is Behavioral Cloning (BC). In BC, the behavior of a skilled human is recorded and, subsequently, an induction algorithm is executed over the traces of the behavior [Bain and Sammut, 1999]. In classical BC, the objective of cloning observed expert demonstrations is defined as a supervised learning problem. This problem is solved by learning a regression function from observed states to actions, i.e., a policy. An impressive early application of BC was the autonomous vehicle ALVINN [Pomerleau, 1989] which learned a neural-network policy for driving a car from recorded state-action training pairs of a human driver. Advantages of BC are the straightforward application and the clean-up effect, i.e., the smoothing of noisy imperfect demonstrations. However, BC is not robust to changes in the control task and the environment. Therefore, it does not have strong performance guarantees.

One main difficulty in imitation learning is the correspondence problem [Nehaniv and Dautenhahn, 2002], i.e., that if the body of the teacher and the robot differ an adequate mapping of the teacher's demonstrations to the robot is non-trivial. The correspondence problem can occur in many different forms. One form is due to different anatomies between the teacher and the robot. As consequence, some demonstrated positions of the teacher may not be reachable for the robot. Another form of the correspondence problem are different dynamics properties between teacher and robot. For example, the robots often have torque limits that cannot reach the same velocity as the teacher. Classical behavioral cloning is very sensitive to the correspondence problem because it directly maps recorded states to recorded actions without taking the anatomy and physics of the robot into account.

In this thesis, we propose a novel probabilistic model-based imitation learning approach that addresses the correspondence problem and allows robots to efficiently acquire new behaviors from expert demonstrations [Englert et al., 2013]. The key idea is to directly match the state trajectory of the robot with the teacher's demonstration. Instead of finding a mapping from states to demonstrated actions, as done in classical behavioral cloning, we learn a robot-specific controller such that the corresponding robot trajectory matches the demonstrated trajectory. This approach is no longer a straight forward supervised learning problem. However, it gives us the advantage that we do not need to record the actions of the expert demonstrations, which gives us the ability to choose from a wider range of demonstration methods (e.g., motion capture). Furthermore, we can use the same teacher demonstrations for teaching multiple different robots.

Our approach exploits a forward model of the robot's dynamics, which allows us to generate predictions of the trajectory when applying the current controller. Using these simulations instead of sampling real robot trajectories reduces the interaction time with the robot. Such data efficient learning saves experimental time and reduces the number of repairs. In the absence of a good model, we propose to learn a forward model using a probabilistic non-parametric Gaussian process [Rasmussen and Williams, 2006]. Learning a forward model is especially suited for robots, where it is difficult to model the robot's dynamics with classical control approaches (e.g., the BioRob™, see Figure 1.1). By using therefore a probabilistic model allows us to take uncertainty about the robot's dynamics into account, which reduces model errors [Schneider, 1997, Atkeson and Santamaría, 1997] that especially occur when only a few samples and no informative prior knowledge is available. Furthermore, we do not need to make potentially unrealistic assumptions (e.g., about rigid body dynamics or friction) which are typically made when learning parametric forward models in robotics.

The rest of the thesis is structured as follows: In Chapter 2 we present our problem statement and provide some background on probabilistic model learning. In Chapter 3 we describe our model-based imitation learning approach, where we use RL methods to learn IL policies. In Chapter 4, we demonstrate the viability of our approach on both simulated and real robot experiments. In the latter case, we successfully learn forward models and controllers for the compliant, biomechanically-inspired manipulator shown in Figure 1.1.

2 Problem Statement and Background

Throughout this thesis, we use the following notation. We denote states by $\boldsymbol{x} \in \mathbb{R}^D$ and actions by $\boldsymbol{u} \in \mathbb{R}^E$, respectively. Furthermore, we define a trajectory $\boldsymbol{\tau}$ as a sequence of states $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ for a fixed time horizon T. Our goal is to learn a state-feedback policy π such that $\boldsymbol{u} = \pi(\boldsymbol{x}, \boldsymbol{\theta})$ with policy parameters $\boldsymbol{\theta}$.

2.1 Problem Statement

As input to our learning algorithm, we assume that the teacher provides n trajectories $\boldsymbol{\tau}_i$ from which the robot should imitate the demonstrated behavior. We use probability distributions over trajectories to represent the expert demonstrations by $p(\boldsymbol{\tau}^{\text{exp}})$ and the robot predictions by $p(\boldsymbol{\tau}^{\pi})$, respectively. We put the robot at a start position that is sampled from an initial distribution $p(\boldsymbol{x}_0)$. Our objective is to find a policy π such that the robot's distribution over predicted trajectories $p(\boldsymbol{\tau}^{\pi})$ matches the distribution over demonstrated trajectories $p(\boldsymbol{\tau}^{\text{exp}})$. Using probability distributions over trajectories allows us to represent uncertainty of the robot's dynamics in a principled way. Furthermore, we can model the variability of the demonstrated trajectories (i.e., a transporting task requires at the pick up position of the object a much higher accuracy, and hence the variance of the demonstrations is smaller there than at other positions).

As a similarity measure between these distributions $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^{\pi})$, we use the Kullback-Leibler (KL) divergence [Solomon, 1959]. Hence, our imitation learning objective is to find a policy such that

$$\pi^* \in \arg\min_{\pi} \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}}) || p(\boldsymbol{\tau}^{\pi})\big)\,, \tag{2.1}$$

where $p(\boldsymbol{\tau}^{\text{exp}})$ is the distribution over the observed demonstrated trajectories and $p(\boldsymbol{\tau}^{\pi})$ is the distribution over the predicted trajectories. We create $p(\boldsymbol{\tau}^{\pi})$ with the internal model of the robot.

2.2 Background on Learning Probabilistic Forward Models

We learn a forward model of the robot's dynamics for doing internal simulations, which is related to the concept that humans rely on internal models for planning, control and learning of their dynamics behavior [Wolpert et al., 1995]. A forward model f maps a state $\boldsymbol{x}_{t-1}$ and action $\boldsymbol{u}_{t-1}$

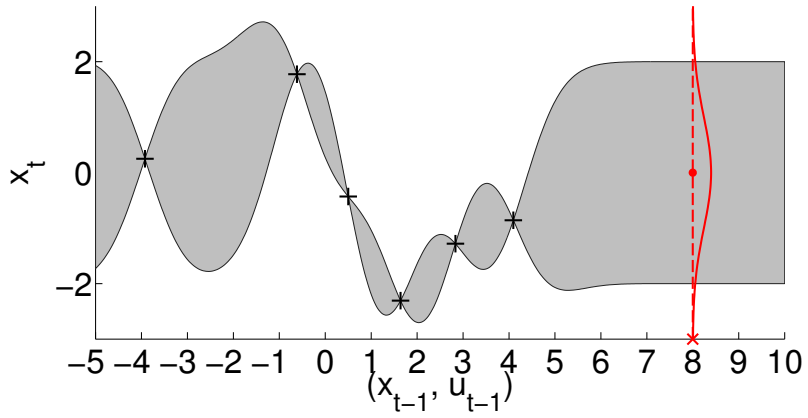


Figure 2.1: Learning a probabilistic forward model with Gaussian processes. The x-axis represents state-action input pairs (x_{t-1}, u_{t-1}) of the model and the y-axis represents the predicted next state x_t. The black plus points denote the training data and the grey shaded area represents two times the standard deviation. It can be seen, that in the region around the training points, our predictions are more certain than at inputs further away. The red line shows a test input for which our model returns the predicted mean and variance.

of the system to the next state $\boldsymbol{x}_t$. In our case, we assume that $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon)$ is i.i.d. Gaussian noise with $\boldsymbol{\Sigma}_\epsilon = \text{diag}([\sigma_1^2 \dots \sigma_D^2])$. Such a model represents the transition dynamics of a robot.

We represent the model by a Gaussian Process (GP), i.e., a probability distribution over models. Since a GP is a consistent, non-parametric method, we can avoid to specify a restrictive parametric model. Instead, a posterior distribution over the underlying function f is inferred directly from the data, while the uncertainty about this estimate is represented as well. As training inputs to the GP, we use state-action pairs $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ and as targets the next state $\boldsymbol{x}_t$. Such a GP represents one-step transitions in the form

$$p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{2.2}$$

$$\text{with} \quad \boldsymbol{\mu}_t = \mathbb{E}_f[f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})] = m_f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}), \tag{2.3}$$

$$\boldsymbol{\Sigma}_t = \text{var}_f[f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})] = \sigma_f^2(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}), \tag{2.4}$$

where m_f is the mean and σ_f^2 the variance of f. An example of such a model is visualized in Figure 2.1.

A GP can be completely specified by a mean function and a covariance function. The mean function allows to incorporate prior knowledge about f and the covariance function defines the

nearness or similarity between inputs. We use a prior mean function $m \equiv 0$ and a squared exponential covariance function plus noise covariance

$$k(\tilde{\boldsymbol{x}}_p, \tilde{\boldsymbol{x}}_q) = \alpha^2 \exp\left(-\tfrac{1}{2}(\tilde{\boldsymbol{x}}_p - \tilde{\boldsymbol{x}}_q)^\top \boldsymbol{\Lambda}^{-1}(\tilde{\boldsymbol{x}}_p - \tilde{\boldsymbol{x}}_q)\right) + \delta_{pq}\boldsymbol{\sigma}_\epsilon^2, \tag{2.5}$$

with inputs of the form $\tilde{\boldsymbol{x}} = [\boldsymbol{x}^\top, \boldsymbol{u}^\top]^\top$. The parameter α^2 is the signal variance, $\boldsymbol{\Lambda} = \text{diag}([l_1^2, \ldots, l_D^2])$ is a matrix with the squared length-scales, and δ_{pq} is the Kronecker symbol, which is 1 when $p = q$, and 0 otherwise.

The posterior predictive distribution at a test input $\tilde{\boldsymbol{x}}_\star$ is given by the mean and variance

$$m_f(\tilde{\boldsymbol{x}}_\star) = \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1}\boldsymbol{y}, \tag{2.6}$$

$$\sigma_f^2(\tilde{\boldsymbol{x}}_\star) = \boldsymbol{k}_{\star\star} - \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1}\boldsymbol{k}_\star \tag{2.7}$$

with $\boldsymbol{k}_\star := k(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{x}}_\star)$, $\boldsymbol{k}_{\star\star} := k(\tilde{\boldsymbol{x}}_\star, \tilde{\boldsymbol{x}}_\star)$, Gram matrix $\boldsymbol{K}$ with $K_{ij} = k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$, and training inputs $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n]$ with corresponding targets $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$. Equations (2.2)–(2.7) are used to simulate the system for a single time-step and map the current state-action pair $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ onto a probability distribution over the next state $\boldsymbol{x}_t$ (see Figure 2.1).

3 Model-based Imitation Learning by Probabilistic Trajectory Matching

Our goal is to imitate the expert's behavior by finding a policy π^* that minimizes the KL divergence between the distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over demonstrated trajectories and the distribution $p(\boldsymbol{\tau}^{\pi})$ over predicted trajectories when executing a policy π, see Equation (2.1).

The KL divergence is a difference measure between two probability distributions and is defined for continuous distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ as

$$\text{KL}\big(p(\boldsymbol{x})||q(\boldsymbol{x})\big) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{3.1}$$

For the special case of two Gaussian distributions $p(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $q(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, the KL divergence has the closed form expression

$$\text{KL}\left(p||q\right) = \tfrac{1}{2}\log\left|\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right| + \tfrac{1}{2}\text{tr}\left(\boldsymbol{\Sigma}_1^{-1}\Big((\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^\top + \boldsymbol{\Sigma}_0 - \boldsymbol{\Sigma}_1\Big)\right). \tag{3.2}$$

We use the KL divergence for our imitation learning approach as a difference measure between probability distributions over trajectories.

3.1 Trajectory Representation

We approximate the distribution over trajectories $p(\boldsymbol{\tau}) = p(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$ by a Gaussian $\mathcal{N}(\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau)$ that factorizes according to

$$p(\boldsymbol{\tau}) \approx \prod_{t=1}^{T} p(\boldsymbol{x}_t) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{3.3}$$

This assumption implies that $\boldsymbol{\Sigma}_\tau \in \mathbb{R}^{TD \times TD}$ is block diagonal without cross-correlations among states at different time steps. In the following paragraphs, we describe how we compute the probability distributions over trajectories $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^{\pi})$ for our objective in Equation (2.1).

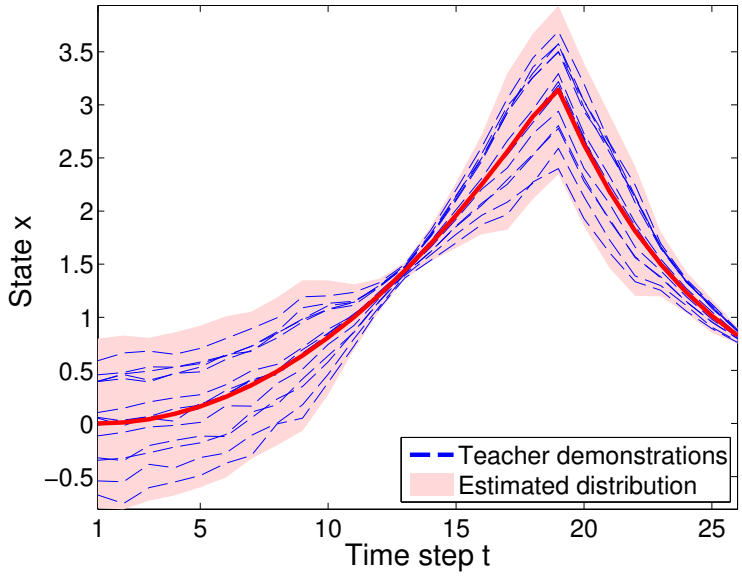


Figure 3.1: Estimation of a Gaussian distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over trajectories (red shaded graph) from multiple expert demonstrations (blue dotted lines).

3.1.1 Estimation of a Distribution over Expert Trajectories

The demonstrations of the teacher are converted such that they are time-aligned, i.e., each trajectory $\boldsymbol{\tau}_i$ consists of a sequence of T states. The mean and covariance matrix of the marginals $p(\boldsymbol{x}_t)$ are computed as unbiased estimates $p(\boldsymbol{x}_t) \approx \mathcal{N}(\hat{\boldsymbol{\mu}}_t^{\text{exp}}, \hat{\boldsymbol{\Sigma}}_t^{\text{exp}})$, where

$$\hat{\boldsymbol{\mu}}_t^{\text{exp}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_t^i, \quad \hat{\boldsymbol{\Sigma}}_t^{\text{exp}} = \frac{1}{n-1}\sum_{i=1}^{n} (\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})(\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})^\top. \tag{3.4}$$

In Equation (3.4), $\boldsymbol{x}_t^i$ is the state after t time steps of the i^{th} demonstrated expert trajectory. This estimation yields an approximate Gaussian distribution over the expert trajectories

$$p(\boldsymbol{\tau}^{\text{exp}}) = \mathcal{N}(\hat{\boldsymbol{\mu}}^{\text{exp}}, \hat{\boldsymbol{\Sigma}}^{\text{exp}}) = \mathcal{N}\left(\begin{bmatrix} \hat{\boldsymbol{\mu}}_1^{\text{exp}} \\ \hat{\boldsymbol{\mu}}_2^{\text{exp}} \\ \vdots \\ \hat{\boldsymbol{\mu}}_T^{\text{exp}} \end{bmatrix}, \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_1^{\text{exp}} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \hat{\boldsymbol{\Sigma}}_2^{\text{exp}} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \hat{\boldsymbol{\Sigma}}_T^{\text{exp}} \end{bmatrix} \right), \tag{3.5}$$

with a block diagonal covariance matrix $\hat{\boldsymbol{\Sigma}}^{\text{exp}}$. An illustration of such a trajectory representation is shown in Figure 3.1. Using Gaussian probability distributions as a representation for the expert demonstrations gives us two advantages: 1) we can exploit the clean-up effect, which washes

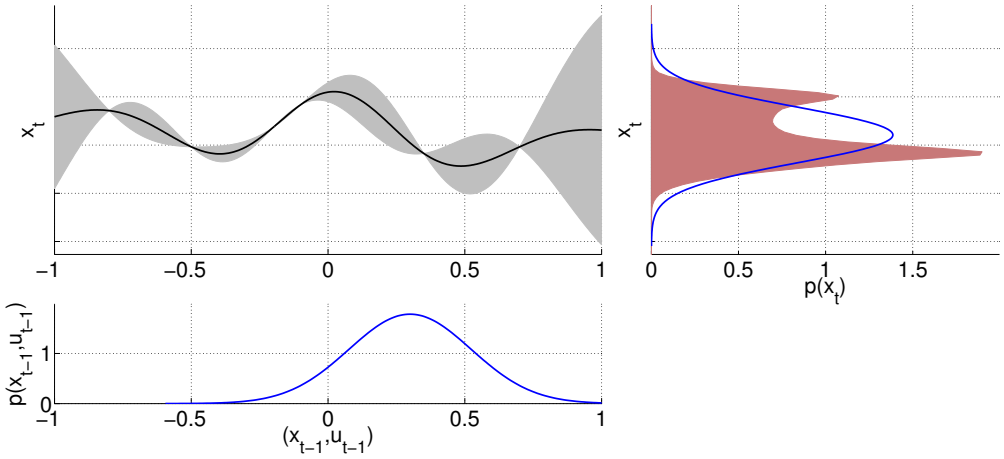


Figure 3.2: Predicting with Gaussian processes at uncertain inputs [Deisenroth and Rasmussen, 2011]. The upper left panel shows the function $f \sim GP$ and the bottom panel shows the Gaussian input distribution $p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1})$. The upper right panel shows the exact prediction $p(\boldsymbol{x}_t)$ shaded in red, which cannot be computed analytically. Therefore, we use exact moment matching to approximate this distribution with a Gaussian $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, which is drawn as a blue line in the upper right panel.

out shaky demonstrations and produces smoother trajectories; 2) we are able to make specific positions more important by representing them with lower variances (e.g., the state at time step 13 in Figure 3.1).

3.1.2 Predicting a Distribution over Robot Trajectories

We use the learned GP forward model described in Section 2.2 for iteratively predicting the state distributions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$ for a given policy π and an initial state distribution $p(\boldsymbol{x}_0)$. These long-term predictions are the marginal distributions of $p(\boldsymbol{\tau}^\pi)$. Note that even for a given input $(\boldsymbol{x}, \boldsymbol{u})$, the GP's prediction is a probability distribution given by Equations (2.6)–(2.7). Iteratively computing the predictions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$, therefore, requires to predict with Gaussian processes at uncertain inputs [Quiñonero-Candela et al., 2003]. The mapping of an uncertain input $p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1})$ through a GP is visualized in Figure 3.2. The input distribution is shown as blue line in the bottom of Figure 3.2 and the GP is shown in the top left of Figure 3.2. The exact predictive distribution $p(\boldsymbol{x}_t)$ is visualized in the top right of Figure 3.2 as red shaded region. As that this distribution is analytically intractable, we approximate it by a Gaussian, which is shown as blue line in the top right of Figure 3.2.

Computing $p(\boldsymbol{x}_t)$ for an uncertain input $p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ requires integrating out both the uncertainty about the state-action pair $\tilde{\boldsymbol{x}}_{t-1}$ and the posterior uncertainty about the function $f \sim GP$ according to

$$p(\boldsymbol{x}_t) = \iint p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1}) p(\tilde{\boldsymbol{x}}_{t-1}) df \, d\tilde{\boldsymbol{x}}_{t-1}. \tag{3.6}$$

The transition probability $p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1})$ is the posterior GP predictive distribution given in Equations (2.6)–(2.7). However, computing the exact distribution $p(\boldsymbol{x}_t)$ in Equation (3.6) is analytically intractable. Therefore, we use exact moment matching and approximate this distribution by a Gaussian as $p(\boldsymbol{x}_t) \approx \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. The mean is computed (using the law of total expectation and Equation (2.6)) by

$$\begin{aligned}
\boldsymbol{\mu}_t &= \mathbb{E}[f(\tilde{\boldsymbol{x}}_{t-1})] = \mathbb{E}_x\left[\mathbb{E}_f[f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1}]\right] \overset{(2.3)}{=} \mathbb{E}_x\left[m_f(\tilde{\boldsymbol{x}}_{t-1})\right] && (3.7)\\
&= \int m_f(\tilde{\boldsymbol{x}}_{t-1}) \mathcal{N}(\tilde{\boldsymbol{x}}_{t-1}|\tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}) d\tilde{\boldsymbol{x}}_{t-1} && (3.8)\\
&\overset{(2.6)}{=} \int k(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{X}}) \mathcal{N}(\tilde{\boldsymbol{x}}_{t-1}|\tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}) d\tilde{\boldsymbol{x}}_{t-1} \boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{\beta}^\top \boldsymbol{q}
\end{aligned}$$

with $\boldsymbol{\beta} = \boldsymbol{K}^{-1}\boldsymbol{y}$ and $\boldsymbol{q} = [q_1, \cdots, q_n]^\top$. Using Equation (2.5), the entries of $\boldsymbol{q}$ are given by

$$\begin{aligned}
q_i &= \int k(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{x}}_i) \mathcal{N}(\tilde{\boldsymbol{x}}_{t-1}|\tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}) d\tilde{\boldsymbol{x}}_{t-1} && (3.9)\\
&= \alpha^2 |\tilde{\boldsymbol{\Sigma}}_{t-1}\boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|^{-\frac{1}{2}} \exp(-\tfrac{1}{2}\boldsymbol{\nu}_i^\top (\tilde{\boldsymbol{\Sigma}}_{t-1} + \boldsymbol{\Lambda})^{-1} \boldsymbol{\nu}_i), && (3.10)
\end{aligned}$$

where we used $\boldsymbol{\nu}_i := \tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{\mu}}_{t-1}$. The predictive covariance matrix $\boldsymbol{\Sigma}_t$ can be derived similarly. The entries of the covariance matrix $\boldsymbol{\Sigma}_t \in \mathbb{R}^{D \times D}$ for the target dimension $a, b = 1, \ldots, D$ are

$$\sigma_{ab}^2 = \boldsymbol{\beta}_a^\top (\boldsymbol{Q} - \boldsymbol{q}_a \boldsymbol{q}_b^\top) \boldsymbol{\beta}_b + \delta_{ab}(\alpha_a^2 - \mathrm{tr}(\boldsymbol{K}^{-1})\boldsymbol{Q}). \tag{3.11}$$

In Equation (3.11), the entries Q_{ij} of $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ are given by

$$\begin{aligned}
Q_{ij} =& \alpha_a^2 \alpha_b^2 \left|(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1})\tilde{\boldsymbol{\Sigma}}_t + \boldsymbol{I}\right|^{-\frac{1}{2}} \times \exp\left(-\tfrac{1}{2}(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)^\top (\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1} (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)\right)\\
&\times \exp\left(-\tfrac{1}{2}(\hat{\boldsymbol{z}}_{ij} - \tilde{\boldsymbol{\mu}}_{t-1})^\top \times \left((\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1})^{-1} + \tilde{\boldsymbol{\Sigma}}_{t-1}\right)^{-1} (\hat{\boldsymbol{z}}_{ij} - \tilde{\boldsymbol{\mu}}_{t-1})\right) && (3.12)
\end{aligned}$$

with

$$\hat{\boldsymbol{z}}_{ij} = \boldsymbol{\Lambda}_b(\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_i + \boldsymbol{\Lambda}_a(\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_j \tag{3.13}$$

where $i, j = 1, \ldots, n$. For a detailed derivation of these results, see [Deisenroth and Rasmussen, 2011].

The GP predictions at uncertain inputs from Equations (3.7)–(3.13) allow the system to iteratively predict the long-term outcome of a control strategy π for a given distribution of the start state $\boldsymbol{x}_0$, which results in a probability distribution over trajectories $p(\boldsymbol{\tau}^\pi)$.

3.2 Natural Cost Function

The trajectory factorization in Equation (3.3) simplifies the KL divergence in Equation (3.1) as it suffices to sum up the KL divergences of the marginal distributions $p(\boldsymbol{x}_t^{\text{exp}})$, $p(\boldsymbol{x}_t^\pi)$ and we obtain the imitation learning objective function

$$J_{\text{IL}}^\pi(\boldsymbol{\theta}) = \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^\pi)\big) = \sum_{t=1}^{T} \text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^\pi)\big)\,. \tag{3.14}$$

Here, we used the trajectory representations from Section 3.1 and Equation (3.1). Since the marginals are approximated by Gaussians, the KL divergence in Equation (3.14) can be evaluated in closed-form by applying Equation (3.2).

Matching the predicted trajectory of the current policy with the expert trajectory via minimizing the KL divergence induces a natural cost function in a standard RL context: Equation (3.14) shows that matching two factorized distributions by means of the KL divergence leads to an additive objective function. Therefore, with $c(\boldsymbol{x}_t) = \text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^\pi)\big)$, we can use reinforcement learning methods to minimize Equation (3.14) since the IL objective J_{IL}^π corresponds to a RL long-term cost J_{RL}^π of the form

$$J_{\text{RL}}^\pi(\boldsymbol{\theta}) = \sum_{t=1}^{T} c(\boldsymbol{x}_t) = \sum_{t=1}^{T} \text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^\pi)\big) = \sum_{t=1}^{T} \text{KL}\big(\mathcal{N}(\hat{\boldsymbol{\mu}}_t^{\text{exp}}, \hat{\boldsymbol{\Sigma}}_t^{\text{exp}})||\mathcal{N}(\boldsymbol{\mu}_t^\pi, \boldsymbol{\Sigma}_t^\pi)\big). \tag{3.15}$$

In Equation (3.15), we used our assumption that trajectories are represented by Gaussian distributions with block-diagonal covariance matrices.

Since $\text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^\pi)\big)$ corresponds to a RL long-term cost function, we can apply RL algorithms to find optimal policies. In principle, any algorithm that can approximate trajectories of

Algorithm 1 Probabilistic Model-based Imitation Learning

input: n expert trajectories $\boldsymbol{\tau}_i$ of a task
init: Estimate expert distribution over trajectories $p(\boldsymbol{\tau}^{\text{exp}})$ (see Section 3.1.1)
Record state-action pairs of the robot (e.g., through applying random control signals)
repeat
Learn probabilistic forward model (GP) for predicting $p(\boldsymbol{\tau}^{\pi})$ (see Section 2.2)
Learn policy parameters $\boldsymbol{\theta}^{\star} \in \arg\min_{\boldsymbol{\theta}} \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^{\pi})\big)$ (see Section 3.3)
Apply $\pi(\boldsymbol{\theta}^{\star})$ to system and record data
until task learned

the current policy π is suitable. For instance, model-free methods based on sampling trajectories directly from the robot [Sutton and Barto, 1998, Peters et al., 2010] or model-based RL algorithms that learn forward models of the robot and, subsequently, use them for predictions [Doya, 2000, Ng and Jordan, 2000, Bagnell and Schneider, 2001, Deisenroth et al., 2011], are suitable. In this thesis, we use a policy search method with learned probabilistic forward models to minimize the KL divergence $\text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^{\pi})\big)$.

3.3 Policy Learning

We use the probabilistic inference for learning control (PILCO) framework [Deisenroth and Rasmussen, 2011] as RL method and adapt it to imitation learning for matching trajectories (an overview of our method is given in Algorithm 1). Our objective is to find policy parameters $\boldsymbol{\theta}$ of a policy π that minimize the long-term cost in Equation (3.15). To find policy parameters $\boldsymbol{\theta}$ such that the distribution over the predicted trajectory matches the distribution over the expert trajectory, we minimize our cost function in (3.14) by means of gradient-based optimization. The gradient of our cost function with respect to the policy parameters $\boldsymbol{\theta}$ is

$$\frac{dJ_{\text{IL}}^{\pi}}{d\boldsymbol{\theta}} = \sum_{t=1}^{T} \left(\frac{\partial \text{KL}}{\partial \boldsymbol{\mu}_t^{\pi}} \frac{d\boldsymbol{\mu}_t^{\pi}}{d\boldsymbol{\theta}} + \frac{\partial \text{KL}}{\partial \boldsymbol{\Sigma}_t^{\pi}} \frac{d\boldsymbol{\Sigma}_t^{\pi}}{d\boldsymbol{\theta}} \right), \tag{3.16}$$

where we require the partial derivatives of the KL divergence with respect to the mean $\boldsymbol{\mu}_t^{\pi}$ and the covariance $\boldsymbol{\Sigma}_t^{\pi}$ of the predicted state distribution at time t. The partial derivatives are given by

$$\frac{\partial \text{KL}}{\partial \boldsymbol{\mu}_t^{\pi}} = -(\boldsymbol{\Sigma}_t^{\pi})^{-1}(\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi}), \tag{3.17}$$

$$\frac{\partial \text{KL}}{\partial \boldsymbol{\Sigma}_t^{\pi}} = \tfrac{1}{2}(\boldsymbol{\Sigma}_t^{\pi})^{-1} - \tfrac{1}{2}(\boldsymbol{\Sigma}_t^{\pi})^{-1}\Big((\boldsymbol{\Sigma}_t^{\pi})^{-1} + (\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi})(\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi})^{\top}\Big)(\boldsymbol{\Sigma}_t^{\pi})^{-1}. \tag{3.18}$$

The derivatives of the mean $\boldsymbol{\mu}_t^\pi$ and covariance $\boldsymbol{\Sigma}_t^\pi$ with respect to $\boldsymbol{\theta}$ are the same as in the regular case [Deisenroth and Rasmussen, 2011]. All the derivatives in Equation (3.16) can be computed analytically and allow the use of fast gradient-based optimization methods such as CG or BFGS.

With the KL divergence as difference measure between the estimated expert distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over trajectories and the predictive distribution $p(\boldsymbol{\tau}^\pi)$ over trajectories, we have formulated model-based imitation learning as a reinforcement learning problem. Thereby the KL divergence serves as an induced natural cost function. The analytic gradients of the loss function allow us to use gradient-based policy search methods. Therefore, we introduced all ingredients for performing probabilistic model-based imitation learning (see Algorithm 1) and solving the problem defined in Equation (2.1).

4 Experimental Results

In the following experiments, we demonstrate the performance of our model-based imitation learning approach in different experiments. First, we learn a swing up task for a simulated double pendulum. Second, we imitate demonstrations provided via kinesthetic teaching with a tendon-driven real BioRob™ with a complex internal dynamics.

4.1 Controller Parametrization

In the following experiments, we use a non-linear Radial Basis Function (RBF) network with axis-aligned Gaussian features ϕ as controller. This policy can be written as

$$\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) \quad \text{with} \tag{4.1}$$

$$\phi_i(\boldsymbol{x}) = \exp\big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_i)^\top \boldsymbol{\Gamma}^{-1}(\boldsymbol{x} - \boldsymbol{c}_i)\big), \tag{4.2}$$

weights w_i, centers $\boldsymbol{c}_i$, and length scales γ_i of each dimension in $\boldsymbol{\Gamma} = \text{diag}(\gamma_1^2, \gamma_2^2, \ldots, \gamma_D^2)$. For taking the action limits $u_{\max}$ into account, we squash the policy $\tilde{\pi}$ through a sinusoidal function to obtain the torque-restricted policy

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}) = u_{\max} \sin\big(\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta})\big). \tag{4.3}$$

Therefore, we can already take the robot's limitations during planning into account, which restricts the policy to the mechanically possible range.

4.2 Double Pendulum

The double pendulum consists of two links and two actuated joints and is mounted on the ground. The system states consisted of joint positions and velocities $\boldsymbol{x} = [q_1, q_2, \dot{q}_1, \dot{q}_2]^\top$; the motor torques served as actions $\boldsymbol{u} = [u_1, u_2]^\top$. The task was to swing up and balance the double pendulum at the inverted position, see Figure 4.1. Each link had a mass $m = 0.5\,\text{kg}$ and the motor torques were limited to the range $[-3, 3]\,\text{Nm}$. We used a low sampling frequency of $10\,\text{Hz}$, a

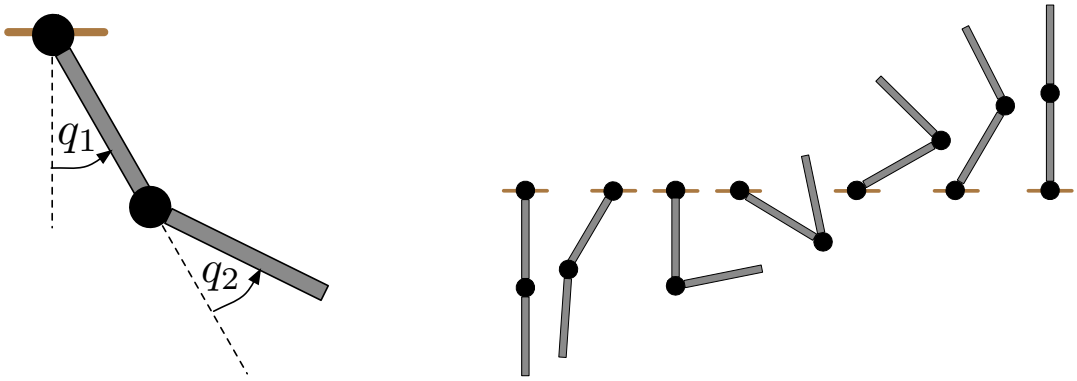


Figure 4.1: Joint angles of the double pendulum (left) and a sequence of positions during the upswing and balancing of the double pendulum (right).

total prediction horizon of $T = 2.5\,\mathrm{s}$, and 100 basis functions ϕ for the RBF network in Equations (4.1)–(4.2). The distribution $p(\boldsymbol{\tau}^{\mathrm{exp}})$ over expert trajectories was based on five successful demonstrations of the task and created according to Section 3.1.1.

4.2.1 Comparison to Reinforcement Learning

To qualitatively evaluate learning success, we compared our proposed model-based IL approach with reinforcement learning. Unlike our IL approach, which matches trajectories by minimizing the KL divergence, reinforcement learning requires a hand-crafted cost function, where we chose the reasonable common cost function

$$c(\boldsymbol{x}) = 1 - \exp(-||\boldsymbol{x} - \boldsymbol{x}_{\mathrm{target}}||^2/\sigma_c^2) \tag{4.4}$$

with the target state $\boldsymbol{x}_{\mathrm{target}}$ and the width scale σ_c^2. As algorithm, we used the same PILCO approach as in our imitation learning framework [Deisenroth and Rasmussen, 2011]. The average success rate as a function of required data is visualized in Figure 4.2. A run was considered successful, when the double pendulum performed the swing-up and balanced in the inverted position. The success rate is given in percent and averaged over ten independent runs of the algorithms. The shaded red graph represents PILCO's learning speed and reaches a success rate of 95 % after about 50 s of robot interactions. The performance of the model-based IL algorithm with random initializations is visualized as the solid blue graph, and reaches a similar success rate after 35 s only. The green dotted line shows also the performance of the model-based IL algorithm, but with an informative initialization of the dynamics model using two expert trajectories. This version achieved the best performance and reached a success rate of around 95 % after only 15 s of experience, which included the two expert trajectories. These results show that the Kullback-Leibler diver-

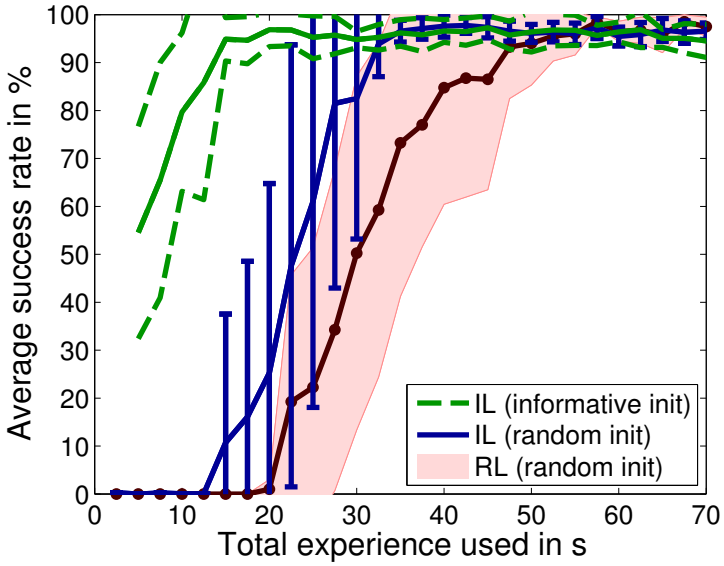


Figure 4.2: Average success rate for solving the double pendulum swing-up task as a function of required data the robot effectively used for learning. The shaded red graph shows the success rate of a learned controller with the hand-crafted reward function from Equation (4.4). The solid blue line shows the performance of the model-based IL approach described in Section 3. Both methods were initialized with a random rollout, hence they start at 2.5 s. The dashed green graph shows the performance of the model-based IL approach, albeit with two expert trajectories as initialization of the dynamics model. We also count these two trajectories as used experience. Thus, the green graph starts at 5 s. All methods are very data efficient and need less than a minute of experience to learn the task.

gence is an appropriate measure for matching trajectory distributions in the context of imitation learning. Furthermore, the integration of expert demonstrations into a cost function can substantially boost the performance in comparison to hand-crafted cost functions. This performance boost can be explained by the fact that the method with the hand-crafted cost function initially needs to explore good trajectories leading to the desired configuration. Our imitation learning variant instead has information about the desired trajectories through the expert's trajectory distribution and, hence, does not rely on excessive exploration. A further speedup can be achieved by initializing the dynamics model learning with informative trajectories, instead of the random initialization, see Figure 4.2.

4.2.2 Addressing the Correspondence Problem

The classical behavioral cloning approach suffers severely from the correspondence problem as it directly uses recorded actions, which makes it sensitive to changes in the body dynamics. Addi-

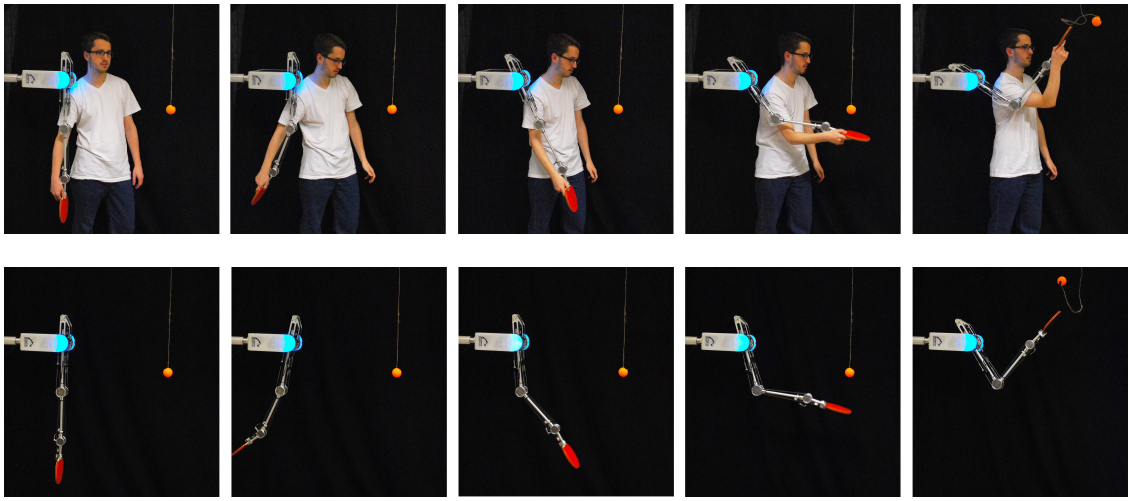


Figure 4.3: The upper photo series shows a kinesthetic demonstration by an expert where just the states are recorded and the lower photos show the movement with the learned controller after the fourth iteration of our approach.

tionally, it restricts the ways how the demonstrations can be recorded. Our approach is more robust to such issues as we train robot-specific controllers. This property gives us the ability to change attributes of the robot (i.e., adding a weight to the robot's end effector) where we are still able to learn. In our experiments with the double pendulum, we have used for generating the demonstration trajectories a robot with different mass values. Instead of the mass $0.5\,\mathrm{kg}$ of each link that the robot had which we used for learning, we generated expert trajectories from a robot which had a mass of $0.8\,\mathrm{kg}$ for the first and $0.6\,\mathrm{kg}$ for the second link. In such a case, classical behavioral cloning fails because the recorded state-actions pairs do not correspond to the robot behavior with the changed attributes. Our approach can still learn a controller as we search for a controller that takes the different attributes during learning into account. Unfortunately, our approach is not robust to all kind of correspondence problems, particularly if the required control commands for imitating the teacher exceed the control limits of the robot. In this case, we can not imitate the teacher, however we may often still be able to find a half-way decent solution.

4.3 Imitation Learning with a BioRob

In the following section, we evaluate the performance of our imitation learning approach on a real robot. We use the biomechanically-inspired compliant BioRob™ [Lens et al., 2010] to learn a fast ball hitting movement, which we demonstrated via kinesthetic teaching. We describe first the robot hardware and the experimental setup, and afterwards we detail model and controller learning.

4.3.1 Hardware Description

The BioRob™ (see Figure 1.1) is a compliant, light-weight robotic arm, capable of achieving high accelerations. Its design tries to place the servo motors close to the torso, minimizing the inertia of the links and enable the end-effector to move with high velocities. Experimental results have shown Cartesian velocities of the end-effector of up to 6.88 m/s [Lens, 2012]. The BioRob X4 is equipped with an end-effector module that increases the total number of degree of freedom to five. The torque is transferred from the motor to the joints via a system of pulleys, drive cables, and springs, which, in the biologically-inspired context, represent tendons and their elasticity. In terms of safety, decoupling the joint and motor inertia protects the items in the robot's workspace and the motor gearboxes in the event of collisions. While the BioRob's design has advantages over traditional approaches, it has the disadvantage that controlling such a compliant system is a highly challenging task.

Classical control approaches that consider only the rigid body dynamics of the system, are unrealistic for controlling the robot, as they omit the cable-driven properties, such as the elasticity of the tendons, the cable slacking effects, stiction, and the energy stored in the cable springs. Linear control approaches suffer even worse form the actuator dynamics. As a result, both forward and inverse dynamics models are not sufficiently accurate for classical control, and the robot fails to follow desired trajectories not even approximately. Moreover, if the control torques are not sufficiently smooth, oscillations close to the eigen-frequency occur. During the oscillations, the motors hold the same position, while the joints, due to the kinematic decoupling from the motors, oscillate. These oscillations differ from the classical under-damped control systems, and, thus, damping them is a non-trivial task.

4.3.2 Task Setup

We attached a table tennis racket to the end-effector of the robot and put a ball on a string hanging down from the ceiling, see Figure 4.3. The shape of the racket alongside with the high velocities produces a significant amount of drag, which is hard to model accurately, leading to substantial errors in parametric models. Thus, learning a non-parametric GP forward model that extracts useful information from data and, subsequently, learning control policies for solving the task, is particularly promising for this compliant tendon-driven robot.

We used three joints of the BioRob™ for performing the task. The state $\boldsymbol{x} \in \mathbb{R}^6$ was given by three joint positions and velocities of the robot; the actions $\boldsymbol{u} \in \mathbb{R}^3$ were given by the corresponding motor torques. The applied motor commands to the robot are the outcome of the policy in Equation (4.3) without any feedback component. We provided three demonstrations of the task

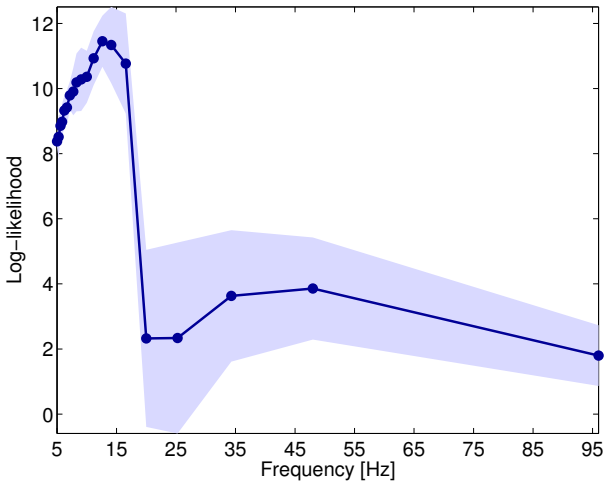


Figure 4.4: Cross-validation of the frequency value f_s. The frequency describes the number of samples taken per second. We model the robot's forward dynamics by a Gaussian process (Equations (2.2)–(2.4)). Therefore, it is important to select a good data sampling frequency in order to model the relevant characteristic of the underlying forward dynamics. We used the log-likelihood (Equation (4.5)) to measure the predictive power of the learned model. The values are averaged over five different training/test folds and plotted with their standard deviation.

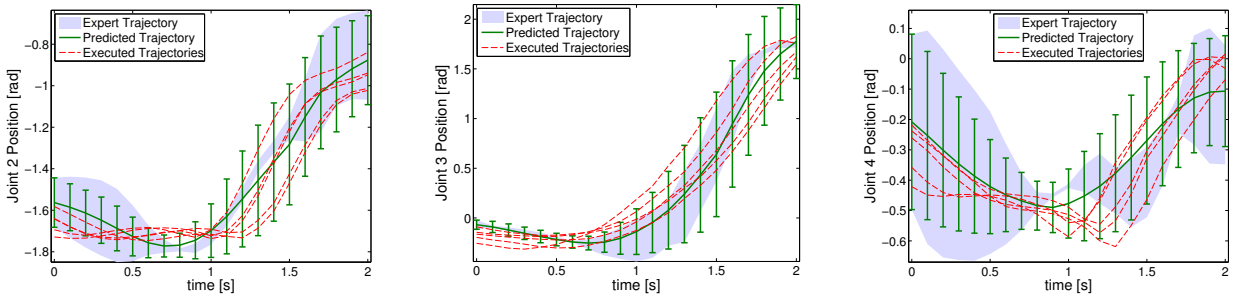
via kinesthetic teaching, as shown in Figure 4.3, to create a distribution over expert trajectories. Therefore, we took the robot by the hand and recorded the system states. The task was first to move back and then to perform a fast up movement to hit the ball, see Figure 4.3.

4.3.3 Model Learning

An important parameter when learning models is the sampling frequency $f_s = 1/\Delta T$ where ΔT is the time difference between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ in our learned forward model, see Equations (2.2)–(2.4). High frequencies result in increased computational time as the number of time steps for a given prediction horizon increases. Moreover, changes in succeeding states can be too insignificant to learn a robust model: Small changes increase the risk that important information about the underlying function is filtered out.

For finding an appropriate sampling frequency f_s, we used k-fold cross-validation with the log-likelihood of our GP predictions. We divided the recorded data into $k = 5$ training/test folds and computed for each fold the predictive log-likelihood with different f_s values. The log-likelihood for one fold is defined as

$$\log p(\boldsymbol{y}_i|\boldsymbol{X}, \boldsymbol{y}_{-i}) = -\tfrac{1}{2}\log|\boldsymbol{\Sigma}_i| - \tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}(\boldsymbol{y}_i - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{\mu}_i). \tag{4.5}$$



(a) Position of Joint 2. (b) Position of Joint 3. (c) Position of Joint 4.

Figure 4.5: Results after learning the imitation of a task with the BioRob™ from kinesthetic teaching. The figures above show the distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over expert trajectories (shaded blue area) and the distribution $p(\boldsymbol{\tau}^{\pi})$ over predicted trajectories from the learned forward model (green error bars). Both are plotted with two times the standard deviation. The red dashed lines show some executed trajectories of the robot where we applied the learned policy. There start state was sampled from the initial distribution $p(\boldsymbol{x}_0)$.

Here $\boldsymbol{y}_{-i}$ denotes the training set without the test values $\boldsymbol{y}_i$ of the current fold, D is the number of test values and $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the predicted mean and variance of the test inputs $\boldsymbol{x}_i$ according to Equations (2.6)–(2.7), respectively. Figure 4.4 shows the averaged log-likelihood over different frequencies f_s. These results show that a sampling frequency f_s of around 10 Hz is most suited for model learning. Higher f_s values reach a lower predictive log-likelihood, which can be explained either by the fact that either they fit to the measurement noise leading to overfitting or the signal-to-noise ratio is very low.

4.3.4 Controller Learning

For learning a BioRob controller we used the RBF network from Equation (4.3) with 80 basis functions and selected a sampling frequency of 10 Hz for our GP forward model according to the results of 4.2.1. We used earlier recorded state-action pairs from simple movements to initialize our forward model, which corresponds to a total experience of 6 s. Our probabilistic IL approach based on trajectory matching led to rapid learning. After the second attempt, the robot was already able to hit the ball and to do a movement similar to the teacher's. After the fourth trial, the robot solved the task and could imitate the demonstrations reliable.

The predicted distribution $p(\boldsymbol{\tau}^{\pi})$ over robot trajectories, the expert distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over demonstrated trajectories, and some executed trajectories after four learning iterations of our proposed IL approach are visualized in Figure 4.5. The figure shows the positions of the three joints that were used for the task. The trajectory prediction of the GP is shown as green error bars. The blue shaded graph is the expert trajectory. Some executed trajectories of the robot where we

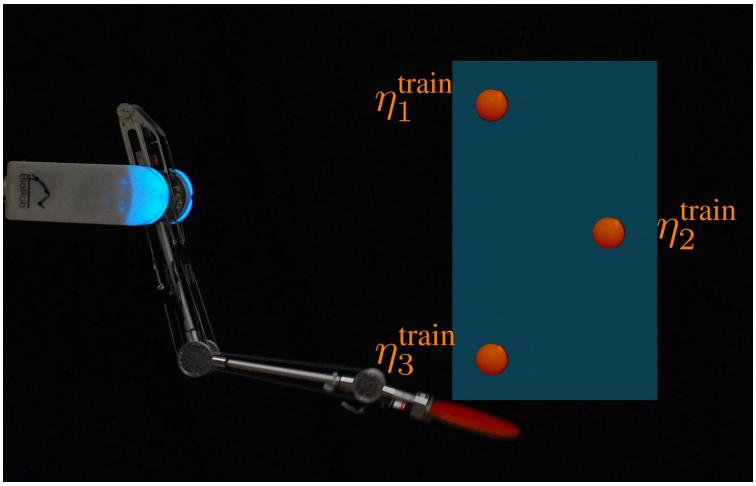


Figure 4.6: Setup for the imitation learning experiments. The orange balls represent the three training targets $\boldsymbol{\eta}_i^{\text{train}}$. The blue rectangle indicates the regions of the test targets $\boldsymbol{\eta}_j^{\text{test}}$ for our learned controller to which we want to generalize.

applied the learned policy are shown as red dashed lines. The robot was able to imitate the demonstrations in a robust manner from different starting positions, using a total of less than 30 s of data to learn both an accurate forward model and a robust controller.

4.3.5 Generalization to Multiple Targets

The policy that we learned for the previous experiments was able to imitate a single task. One of the key challenges in imitation learning is the generalization to more complex tasks, where a policy needs to be adaptable to changes in the environment (e.g., an object position).

For making the learned policy able to generalize to such changes, we incorporate our IL framework into the multi target scenario [Deisenroth and Fox, 2011, Deisenroth et al., 2013]. Therefore, we added target variables $\boldsymbol{\eta}$ as input to our policy π. The key idea is now that we train our policy $\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\eta})$ on a small predefined set of training targets $\boldsymbol{\eta}_i^{\text{train}}$. Then in the test phase we expect to generalize to previously unseen, but related, test targets $\boldsymbol{\eta}_i^{\text{test}}$. The generalization ability is achieved through the representation of the training targets as Gaussian distributions. Therefore, the controller automatically learns the implicit similarity of the targets. For a detailed description of this approach, we refer to [Deisenroth et al., 2013].

Our multiple target loss function is now defined as

$$J_{\text{MT}}^{\pi}(\boldsymbol{\theta}, \boldsymbol{\eta}) \approx \frac{1}{M} \sum_{i=1}^{M} J_{\text{IL}}^{\pi}\left(\boldsymbol{\theta}, \boldsymbol{\eta}_i^{\text{train}}\right) = \frac{1}{M} \sum_{i=1}^{M} \text{KL}\left(p(\boldsymbol{\tau}_i^{\text{exp}}) || p(\boldsymbol{\tau}^{\pi} | \boldsymbol{\eta}_i^{\text{train}})\right), \tag{4.6}$$

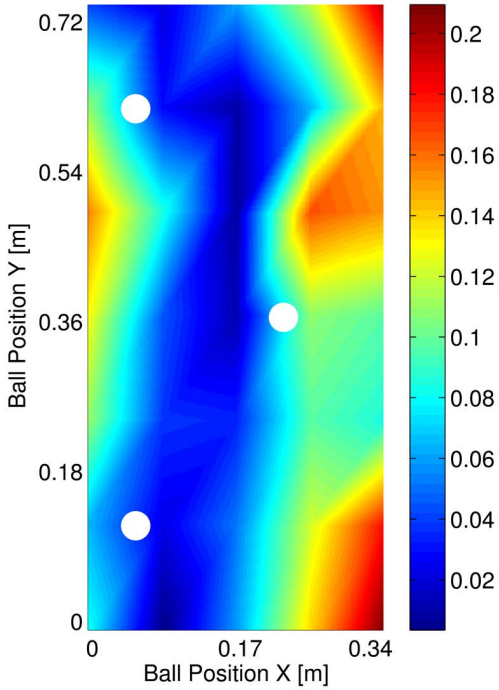


Figure 4.7: Evaluation of the imitation learning experiments with the BioRob™. The three white discs show the training target locations. The color represents the minimum distance between the ball position and the center of the table-tennis racket trajectory.

where we sum the loss function J_{IL}^{π} from Equation (2.1) over M training targets $\boldsymbol{\eta}_i^{\text{train}}$. For each training target we have to demonstrate one corresponding expert trajectory distribution $p(\boldsymbol{\tau}_i^{\text{exp}})$. Optimizing Equation (4.6) with Algorithm 1 allows us to learn a single policy, which is now an explicit function of the target $\boldsymbol{\eta}$.

We evaluated this multiple target imitation learning by extending the experiments from Section 4.3.4 to variable ball positions in a 2D-plane. Therefore, we used the RBF network of Equation (4.3) with 250 Gaussian basis functions, where the policy parameters comprised the locations of the basis functions, their weights, and a shared diagonal covariance matrix, resulting in approximately 2300 policy parameters.

A target was represented as a two-dimensional vector $\boldsymbol{\eta} \in \mathbb{R}^2$ corresponding to the ball position in Cartesian coordinates in an arbitrary reference frame within the hitting plane. As training targets $\boldsymbol{\eta}_j^{\text{train}}$, we defined hitting movements for three different ball positions (see Figure 4.6). For each training target, an expert demonstrated two hitting movements via kinesthetic teaching. Our goal was to learn a policy that a) learns to imitate three distinct expert demonstrations, and b) generalizes from demonstrated behaviors to targets that were not demonstrated. In particular, these test targets were to hit balls in a larger region around the training locations, indicated by the blue box in Figure 4.6. Figure 4.7 shows the performance results as heatmap after 15 iterations of Algorithm 1. The evaluation measure was the distance between the ball position and the center of the table-tennis

racket. We computed this error in a regular 7x5 grid of the blue area in Figure 4.7. The distances in the blue and cyan areas were sufficient to successfully hit the ball. We successfully generalized demonstrations to new targets.

5 Conclusion

In this thesis, we have presented a probabilistic model-based imitation learning approach which enables robots to acquire new tricks through teacher demonstrations. The three key components of our approach are: 1) the probabilistic modeling of the robot's dynamics and the teacher's demonstrations, that allows us to take uncertainty appropriately into account; 2) the mental rehearsal of the current controller with predictions of distributions over plausible trajectories guarantees data efficient learning; 3) the search for robot-specific controllers that match the robot trajectory with the expert trajectory, which enables us to use demonstration methods that do not record the actions of the teacher. We have shown that matching trajectories with the Kullback-Leibler divergence as similarity measure is equivalent to a reinforcement learning problem, where the similarity measure serves as an induced immediate cost function. The experiments have shown that our approach provides a fast learning and addresses the correspondence problem, which allows us to be robust to changes in the task setup. Furthermore, we demonstrated the applicability of our approach to real robots, where we used a compliant robot to imitate demonstrations provided by kinesthetic teaching in a fast and robust manner.

List of Figures

Bibliography

[Abbeel et al., 2008] Abbeel, P., Dolgov, D., Ng, A., and Thrun, S. (2008). Apprenticeship Learning for Motion Planning with Application to Parking Lot Navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Argall et al., 2009] Argall, B., Chernova, S., Veloso, M., and Browning, B. (2009). A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*.

[Atkeson and Santamaría, 1997] Atkeson, C. and Santamaría, J. C. (1997). A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the International Conference on Robotics and Automation*.

[Atkeson and Schaal, 1997] Atkeson, C. and Schaal, S. (1997). Robot Learning from Demonstration. In *Proceedings of the International Conference on Machine Learning*.

[Bagnell and Schneider, 2001] Bagnell, J. and Schneider, J. (2001). Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *Proceedings of the International Conference on Robotics and Automation*.

[Bain and Sammut, 1999] Bain, M. and Sammut, C. (1999). A Framework for Behavioural Cloning. *Machine Intelligence*.

[Ben Amor et al., 2009] Ben Amor, H., Berger, E., Vogt, D., and Jung, B. (2009). Kinesthetic Bootstrapping: Teaching Motor Skills to Humanoid Robots through Physical Interaction. *KI 2009: Advances in Artificial Intelligence*, pages 492–499.

[Biggs and Macdonald, 2003] Biggs, G. and Macdonald, B. (2003). A Survey of Robot Programming Systems. In *Proceedings of the Australian Conference on Robotics and Automation*.

[Boularias et al., 2011] Boularias, A., Kober, J., and Peters, J. (2011). Relative Entropy Inverse Reinforcement Learning. *In Proceedings of AISTATS*.

[Calinon et al., 2007] Calinon, S., Guenter, F., and Billard, A. (2007). On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics*.

[Deisenroth et al., 2013] Deisenroth, M., Englert, P., Peters, J., and Fox, D. (2013). Multi-Task Policy Search for Reinforcement Learning and Robotics. *Under review by International Conference on Machine Learning 2013*.

[Deisenroth and Fox, 2011] Deisenroth, M. and Fox, D. (2011). Multiple-Target Reinforcement Learning with a Single Policy. *In ICML 2011 Workshop on Planning and Acting with Uncertain Models*.

[Deisenroth and Rasmussen, 2011] Deisenroth, M. and Rasmussen, C. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. *Proceedings of the International Conference on Machine Learning*.

[Deisenroth et al., 2011] Deisenroth, M., Rasmussen, C., and Fox, D. (2011). Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science & Systems*.

[Doya, 2000] Doya, K. (2000). Reinforcement Learning in Continuous Time and Space. *Neural Computation*.

[Englert et al., 2013] Englert, P., Paraschos, A., and Peters, J. Deisenroth, M. (2013). Behavioral cloning with learned probabilistic forward models. *Proceedings of the International Conference on Robotics and Automation*.

[Kober and Peters, 2010] Kober, J. and Peters, J. (2010). Imitation and Reinforcement Learning. *IEEE Robotics & Automation Magazine*.

[Lens, 2012] Lens, T. (2012). *Physical Human-Robot Interaction with a Lightweight, Elastic Tendon Driven Robotic Arm: Modeling, Control, and Safety Analysis*. PhD thesis, TU Darmstadt, Department of Computer Science.

[Lens et al., 2010] Lens, T., Kunz, J., Stryk, O. v., Trommer, C., and Karguth, A. (2010). BioRob-Arm: A Quickly Deployable and Intrinsically Safe, Light-Weight Robot Arm for Service Robotics Applications. *International Symposium on Robotics*.

[Nehaniv and Dautenhahn, 2002] Nehaniv, C. and Dautenhahn, K. (2002). The Correspondence Problem. *In Imitation in Animals and Artifacts*.

[Ng and Jordan, 2000] Ng, A. and Jordan, M. (2000). PEGASUS: A Policy Search Method for Large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*.

[Ng et al., 2003] Ng, A., Kim, H., Jordan, M., and Sastry, S. (2003). Autonomous Helicopter Flight Via Reinforcement Learning. In *Proceedings of the International Symposium on Experimental Robotics*.

[Ng and Russell, 2000] Ng, A. and Russell, S. (2000). Algorithms for Inverse Reinforcement Learning. *Proceedings of the International Conference on Machine Learning*.

[Peters et al., 2010] Peters, J., Mülling, K., and Altun, Y. (2010). Relative Entropy Policy Search. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*.

[Pomerleau, 1989] Pomerleau, D. (1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. *Advances in Neural Information Processing System*.

[Quiñonero-Candela et al., 2003] Quiñonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. (2003). Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[Rasmussen and Williams, 2006] Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[Schneider, 1997] Schneider, J. (1997). Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. In *Advances in Neural Information Processing Systems*. Morgan Kaufman Publishers.

[Solomon, 1959] Solomon, K. (1959). *Information Theory and Statistics*. Wiley, New York.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Ude et al., 2004] Ude, A., Atkeson, C., and Riley, M. (2004). Programming Full-Body Movements for Humanoid Robots by Observation. *Robotics and Autonomous Systems*.

[Wolpert et al., 1995] Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An Internal Model for Sensorimotor Integration. *Science, 269*.

[Zöllner et al., 2004] Zöllner, R., Asfour, T., and Dillmann, R. (2004). Programming by Demonstration: Dual-Arm Manipulation Tasks for Humanoid Robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*.