Model-based Imitation Learning by Probabilistic Trajectory Matching

Peter Englert[1], Alexandros Paraschos[1], Jan Peters[1,2], Marc Peter Deisenroth[1]

***Abstract*— One of the most elegant ways of teaching new skills to robots is to provide demonstrations of a task and let the robot imitate this behavior. Such imitation learning is a non-trivial task: Different anatomies of robot and teacher, and reduced robustness towards changes in the control task are two major difficulties in imitation learning. We present an imitation-learning approach to efficiently learn a task from expert demonstrations. Instead of finding policies indirectly, either via state-action mappings (behavioral cloning), or cost function learning (inverse reinforcement learning), our goal is to find policies directly such that predicted trajectories match observed ones. To achieve this aim, we model the trajectory of the teacher and the predicted robot trajectory by means of probability distributions. We match these distributions by minimizing their Kullback-Leibler divergence. In this paper, we propose to learn probabilistic forward models to compute a probability distribution over trajectories. We compare our approach to model-based reinforcement learning methods with hand-crafted cost functions. Finally, we evaluate our method with experiments on a real compliant robot.**

I. Introduction

Instructing robots to perform complex tasks is essential for using them in industrial or daily life situations. Classical methods of instructing robots are programming methods like offline programming with simulated systems [13] or online programming through teach-in [7]. However, these methods suffer from the large amount of work needed for teaching a single task and the difficulty of transferring programs to new environments without starting from scratch.

For complex tasks, it is often impractical to program robot behavior by hand. Thus, imitation learning aims at transferring skills from a teacher via demonstrations to a robot [4], [2]. Providing human demonstrations and letting the robot imitate this behavior is often easier than manually programming controllers. Research in the field of imitation learning has led to a variety of different techniques. They differ in the way the demonstrations are provided (e.g., motion capture [28], kinesthetic teaching [8]), the level at which imitation happens (e.g., at the symbolic [29] or the trajectory level [9]), whether they use a system model, and whether/how they employ reward functions for the task.

A common technique to improve the results of imitation learning is Reinforcement Learning (RL). In RL, a task-specific reward function is defined, which is maximized [27]. Often the results of imitation learning are used as an initialization to RL. This approach was used to make a robot learn the ball-in-a-cup game, where the learning was initialized by human demonstrations [14]. However, a major difficulty in RL is the definition of a suitable reward function for more complex tasks, e.g., when performing a tennis stroke.

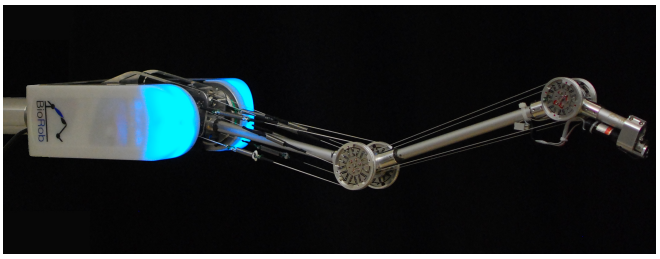


Fig. 1. The BioRob^{TM} is a compliant, biomechanically-inspired robot manipulator with drive cables and springs, which represent tendons and their elasticity.

Inverse Reinforcement Learning (IRL) addresses the problem of hand-crafting task-specific reward functions [19]. IRL automatically extracts a reward function from demonstrated behavior, which is subsequently used in an RL framework for policy learning. Hence, IRL is suited for tasks where it is difficult to provide a reward function by hand, e.g, flying acrobatic helicopter maneuvers [1], [18]. A drawback of IRL is that the performance relies on feature selection, which can strongly bias the performance [19].

One of the classic forms 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 [6]. In classical BC, the objective is to match observed expert trajectories and robot trajectories indirectly 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 [21]. It learned a neural-network policy for driving a car from recorded state-action training pairs of a human driver. BC is straightforwardly applicable and leads to a clean-up effect, i.e., the smoothing of noisy demonstrations. However, BC is not robust to changes in the control task and cannot provide strong performance guarantees. Moreover, BC suffers severely from the correspondence problem [17], where a direct mapping between the different anatomies of teacher and robot is non-trivial, e.g., if the human demonstrations exceed the torque limits of the robot's motors.

In this paper, we propose a novel approach to imitation learning by probabilistic trajectory matching. The key idea is to find a robot-specific policy such that the observed expert trajectories and the predicted robot trajectories match. We propose to learn the policy not as a mapping from

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement #270327.

[1] Dept. of Computer Science, Technische Universität Darmstadt, Germany.
[2] Max Planck Institute for Intelligent Systems, Tübingen, Germany.
{englert|paraschos|peters|marc}@ias.tu-darmstadt.de

demonstrated states to demonstrated actions, but rather as a function from robot states to robot-specific actions, with the sole goal of matching demonstrated and robot trajectories.

To predict the robot's trajectories, we propose to learn a forward model. To be robust to model errors, our learned forward model is a probability distribution over forward models and implemented as a probabilistic non-parametric Gaussian process (GP) [24]. The GP takes uncertainties about the learned robot dynamics into account and reduces typical problems of learned models, such as model errors [25], [3] and potentially unrealistic assumptions (e.g., rigid body dynamics), typically made in parametric forward models.

Throughout this paper, we use the following notation: States are denoted by $\boldsymbol{x} \in \mathbb{R}^D$ and actions as $\boldsymbol{u} \in \mathbb{R}^E$, respectively. Furthermore, a trajectory as $\boldsymbol{\tau}$ comprises a sequence of states $\boldsymbol{x}_0, \dots, \boldsymbol{x}_T$ for a fixed time horizon T. The policy π maps a state $\boldsymbol{x}$ to a corresponding action $\boldsymbol{u}$.

In the context of imitation learning, our objective is to find a policy π, such that the robot's predicted trajectory $\boldsymbol{\tau}^\pi$ matches the observed expert trajectory $\boldsymbol{\tau}^{\text{exp}}$. We use probability distributions over trajectories for representing both the demonstrated trajectories and the predicted trajectory. Probability distributions over trajectories allows us to represent both the uncertainty about the robot's dynamics and the variability of the demonstrations in a principled way.

As a similarity measure between these distributions we use the Kullback-Leibler (KL) divergence [26]. Hence, our objective is to find a policy π^* with

$$\pi^* \in \arg\min_\pi \mathrm{KL}\big(p(\boldsymbol{\tau}^{\text{exp}}) || p(\boldsymbol{\tau}^\pi)\big)\,, \tag{1}$$

where $p(\boldsymbol{\tau}^\pi)$ is the predicted robot trajectory distribution and $p(\boldsymbol{\tau}^{\text{exp}})$ is the observed expert trajectory distribution.

II. Trajectory Matching

Our goal is to imitate the expert's behavior by minimizing the KL divergence between the distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over demonstrated trajectories and the predicted trajectory distribution $p(\boldsymbol{\tau}^\pi)$ of the robot when executing a policy π, see Eq. (1). In this section, we show that minimizing the KL divergence between these two trajectory distributions induces a natural cost function, that can be used by any RL algorithm to learn policies.

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

$$\mathrm{KL}\big(p(\boldsymbol{x}) || q(\boldsymbol{x})\big) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{2}$$

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 is given by the closed-form expression

$$\begin{aligned}\mathrm{KL}\left(p||q\right) = \tfrac{1}{2} \log \left|\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right| \\ + \tfrac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\big((\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^\top + \boldsymbol{\Sigma}_0 - \boldsymbol{\Sigma}_1\big)\right).\end{aligned} \tag{3}$$

A. Trajectory Representation

We assume that a distribution over trajectories $p(\boldsymbol{\tau}) = p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$ is approximated by a Gaussian $\mathcal{N}(\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau)$ and factorizes according to

$$p(\boldsymbol{\tau}) \approx \prod_{t=1}^T p(\boldsymbol{x}_t) = \prod_{t=1}^T \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)\,. \tag{4}$$

The factorizing assumption implies that $\boldsymbol{\Sigma}_\tau$ is block diagonal without cross-correlations among states at different time steps. An illustration of such a trajectory representation is shown in Fig. 2. We assume that an expert provides n trajectories to the imitation learner. We consider the case that a demonstrated trajectory $\boldsymbol{\tau}_i^{\text{exp}}$ consists of a sequence of states $\boldsymbol{x}_t^i$ for each time step $t = 1, \dots, T$, with a fixed time horizon T. The mean and covariance matrix of the marginals are computed as unbiased estimates

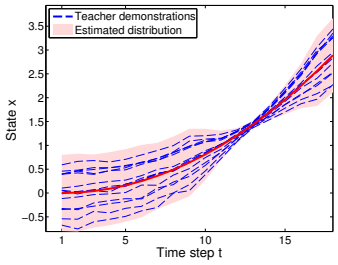


Fig. 2. Teacher demonstrations (blue lines) and estimated expert trajectory distribution (red graph).

$$\hat{\boldsymbol{\mu}}_t^{\text{exp}} = \tfrac{1}{n}\sum_{i=1}^n \boldsymbol{x}_t^i,\; \hat{\boldsymbol{\Sigma}}_t^{\text{exp}} = \tfrac{1}{n-1}\sum_{i=1}^n (\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})(\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})^\top,$$

respectively, where $\boldsymbol{x}_t^i$ is the state at time t of the i^{th} expert trajectory. This yields an approximate Gaussian distribution $p(\boldsymbol{\tau}^{\text{exp}}) = \mathcal{N}(\boldsymbol{\mu}^{\text{exp}}, \boldsymbol{\Sigma}^{\text{exp}})$ over the expert trajectories with

$$\boldsymbol{\mu}^{\text{exp}} = [\hat{\boldsymbol{\mu}}_1^{\text{exp}}, \hat{\boldsymbol{\mu}}_2^{\text{exp}}, \dots, \hat{\boldsymbol{\mu}}_T^{\text{exp}}]^\top \tag{5}$$

$$\text{and } \boldsymbol{\Sigma}^{\text{exp}} = \mathrm{diag}(\hat{\boldsymbol{\Sigma}}_1^{\text{exp}}, \hat{\boldsymbol{\Sigma}}_2^{\text{exp}}, \dots, \hat{\boldsymbol{\Sigma}}_T^{\text{exp}}). \tag{6}$$

The trajectory factorization in Eq. (4) simplifies the KL divergence in Eq. (2) as it suffices to sum up the KL divergences of the marginal distributions $p(\boldsymbol{x}_t)$, $q(\boldsymbol{x}_t)$, i.e.,

$$\mathrm{KL}\big(p(\boldsymbol{\tau})||q(\boldsymbol{\tau})\big) = \sum_{t=1}^T \mathrm{KL}\big(p(\boldsymbol{x}_t)||q(\boldsymbol{x}_t)\big). \tag{7}$$

Since the marginals $p(\boldsymbol{x}_t)$ and $q(\boldsymbol{x}_t)$ are approximated by Gaussians, the KL divergence in Eq. (7) can be evaluated in closed-form using Eq. (3).

B. Natural Cost Function

Matching the predicted trajectory of the current policy $p(\boldsymbol{\tau}^\pi)$ with the expert trajectory distribution $p(\boldsymbol{\tau}^{\text{exp}})$ via minimizing the KL divergence induces a natural cost function in a standard RL context: Eq. (7) shows that matching two factorized distributions by means of the KL divergence leads to an additive objective function. More specifically, for the trajectory distributions $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^\pi)$, we aim to find a policy π that minimizes the objective

$$J_{\text{IL}}^\pi = \mathrm{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^\pi)\big) = \sum_{t=1}^T \mathrm{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^\pi)\big). \tag{8}$$

For finding a policy that minimizes J_{IL}^π in Eq. (8), we can use standard RL methods: Our objective in Eq. (8) corresponds to an RL long-term cost of the form

$$J_{\text{RL}}^\pi = \sum_{t=1}^T c(\boldsymbol{x}_t) \tag{9}$$

Algorithm 1 PILCO

init: Apply random control signals and record data
repeat
  Learn probabilistic forward model (GP)
  Policy Search
  repeat
    Approximate inference for policy evaluation
    Gradient-based policy improvement $dJ^\pi(\boldsymbol{\theta})/d\boldsymbol{\theta}$
    Update parameter $\boldsymbol{\theta}$
  until convergence **return** $\boldsymbol{\theta}^\star$
  Set $\pi^\star \leftarrow \pi(\boldsymbol{\theta}^\star)$
  Apply $\pi^\star$ to system and record data
until task learned

with an immediate cost $c(\boldsymbol{x}_t) = \mathrm{KL}\big(p(\boldsymbol{x}_t^{\mathrm{exp}})||p(\boldsymbol{x}_t^\pi)\big)$.

Since the KL divergence between trajectory distributions in Eq. (7) corresponds to a RL long-term cost function, see Eq. (9), we can apply RL algorithms to find optimal policies. In principle, any algorithm that can approximate trajectories is suitable. For instance, model-free methods based on trajectory sampling [27], [20] or model-based RL algorithms that learn forward models of the robot, and, subsequently, use them for predictions [12], [5], [23], [11], are suitable. In this paper, we use a policy search method with learned probabilistic forward models to minimize the KL divergence $\mathrm{KL}\big(p(\boldsymbol{\tau}^{\mathrm{exp}})||p(\boldsymbol{\tau}^\pi)\big)$.

III. Imitation Learning via Policy Search

For performing policy search, we use the PILCO (probabilistic inference for learning control) framework [10], which learns policies data efficiently. An outline of PILCO is given in Algorithm 1.

The objective in policy search is to find policy parameters $\boldsymbol{\theta}$ of a policy π that minimize the long-term cost in Eq. (9). PILCO possesses two key ingredients to find good policy parameters: 1) learning of a probabilistic forward model of the robot dynamics with a GP; 2) analytic long-term predictions and policy gradients $\partial J_{\mathrm{RL}}^\pi/\partial\boldsymbol{\theta}$. PILCO explicitly accounts for the uncertainty about the unknown dynamics function long-term predictions and controller learning. Hence, learning is relatively robust to modeling errors.

RL algorithms require an immediate cost function c in Eq. (9), the specification of which can be very cumbersome: For complex tasks it is non-trivial to find a suitable function that allows learning the task properly. However, in the imitation-learning context of this paper, a natural RL cost function c is induced by matching trajectory distributions by minimizing the KL divergence between them, see Eq. (9). We will exploit this natural cost function in the following to find imitation-learning policies.

To find policy parameters $\boldsymbol{\theta}$ such that the predicted trajectory distribution matches the observed expert trajectory distribution, PILCO minimizes the cost function in Eq. (8) by means of gradient-based optimization. For this purpose, PILCO computes the derivatives of J_{IL}^π with respect to the policy parameters $\boldsymbol{\theta}$

$$\frac{dJ_{\mathrm{IL}}^\pi}{d\boldsymbol{\theta}} = \sum\nolimits_{t=1}^{T} \left(\frac{\partial \mathrm{KL}}{\partial \boldsymbol{\mu}_t^\pi} \frac{d\boldsymbol{\mu}_t^\pi}{d\boldsymbol{\theta}} + \frac{\partial \mathrm{KL}}{\partial \boldsymbol{\Sigma}_t^\pi} \frac{d\boldsymbol{\Sigma}_t^\pi}{d\boldsymbol{\theta}} \right). \tag{10}$$

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 can be computed analytically from Eq. (3). The derivatives of the mean $\boldsymbol{\mu}_t^\pi$ and covariance $\boldsymbol{\Sigma}_t^\pi$ with respect to $\boldsymbol{\theta}$ can be found in [10].

Instead of initializing the model randomly (see Alg. 1), it is also possible to initialize the model with recorded state-action pairs to increase convergence speed. All derivatives in Eq. (10) are computed analytically, which allows for fast gradient-based optimization methods, e.g., BFGS.

So far, we tacitly assumed that the predicted trajectories $\boldsymbol{\tau}^\pi$ are given. In the following sections, we describe how we compute them. First, we learn a probabilistic forward model with GPs. Subsequently, we detail how PILCO computes distributions over long-term trajectories, which are required to imitate the behavior of the demonstrations via minimizing the KL divergence in Eq. (8).

A. Learning a Probabilistic Forward Model

A forward model maps a state $\boldsymbol{x}_{t-1}$ and action $\boldsymbol{u}_{t-1}$ of the system to the next state $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$. Such a model can be used to represent the transition dynamics of a robot. We represent the model by a probability distribution over models and implemented as a GP. Since a GP is a consistent, non-parametric method, it is not necessary to specify restrictive parametric models. Instead, the "shape" of the underlying function f is inferred directly from the data, while the uncertainty about this estimate is represented appropriately as well. As training inputs to the GP we used state-action pairs $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ and as targets the differences of the states $\boldsymbol{\Delta}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_\epsilon)$ is i.i.d. Gaussian noise with $\boldsymbol{\Sigma}_\epsilon = \mathrm{diag}([\boldsymbol{\sigma}_1^2 \dots \boldsymbol{\sigma}_D^2])$. This GP represents one-step predictions in the form

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{11}$$

$$\text{with} \quad \boldsymbol{\mu}_t = \boldsymbol{x}_{t-1} + \mathbb{E}_f[\boldsymbol{\Delta}_t], \tag{12}$$

$$\boldsymbol{\Sigma}_t = \mathrm{var}_f[\boldsymbol{\Delta}_t]. \tag{13}$$

We use a prior mean function $m \equiv 0$ and a squared exponential covariance function plus noise covariance

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

with inputs of the form $\tilde{\boldsymbol{x}} = [\boldsymbol{x}^\top, \boldsymbol{u}^\top]^\top$. The parameter α^2 is the signal variance, $\boldsymbol{\Lambda} = \mathrm{diag}([l_1^2, \dots, l_D^2])$ is a matrix with the squared length-scales, δ_{pq} is the Kronecker symbol which is 1 when $p = q$, and 0 otherwise. The posterior predictive distribution of a deterministic test input $\tilde{\boldsymbol{x}}_\star$ is given by the mean and variance

$$m_f(\tilde{\boldsymbol{x}}_\star) = \mathbb{E}_f[\boldsymbol{\Delta}_\star] = \boldsymbol{k}_\star^T (\boldsymbol{K} + \boldsymbol{\sigma}_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \tag{14}$$

$$\sigma_f^2(\tilde{\boldsymbol{x}}_\star) = \mathrm{var}_f[\boldsymbol{\Delta}_\star] = \boldsymbol{k}_{\star\star} - \boldsymbol{k}_\star^T (\boldsymbol{K} + \boldsymbol{\sigma}_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{k}_\star, \tag{15}$$

where $\boldsymbol{k}_\star := k(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{x}}_\star)$, $\boldsymbol{k}_{\star\star} := k(\tilde{\boldsymbol{x}}_\star, \tilde{\boldsymbol{x}}_\star)$, Gram matrix $\boldsymbol{K}$ with $K_{ij} = k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$, training inputs $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_n]$,

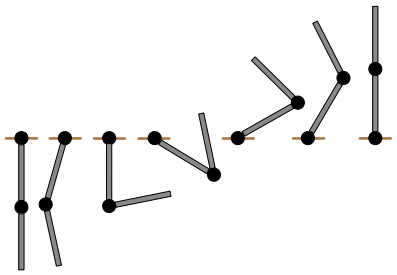


Fig. 3. Swing-up steps of the double-pendulum from the initial to the final configuration. The double pendulum is mounted fixed on the ground and consists of two actuated joints (black circles) and two links (grey bars).

and corresponding targets $\boldsymbol{y} = [\boldsymbol{\Delta}_1, \ldots, \boldsymbol{\Delta}_n]^\top$. Eqs. (11)–(15) are used to map the current state-action pair $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ onto a probability distribution over the next state $\boldsymbol{x}_{t+1}$.

B. Trajectory Predictions

Based on the PILCO framework, we use the learned GP forward model 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 the distribution $p(\boldsymbol{\tau}^\pi)$ over predicted trajectories. However, even for a given input $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, the GP's prediction is a probability distribution given by Eqs. (14)–(15). Iteratively computing the predictions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$, therefore, requires to predict with GPs at uncertain inputs [22]. The predicted distribution at an uncertain input $\tilde{\boldsymbol{x}}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})$ 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

$$p(\boldsymbol{\Delta}_t) = \iint p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1}) p(\tilde{\boldsymbol{x}}_{t-1}) df \, d\tilde{\boldsymbol{x}}_{t-1}. \quad (16)$$

The transition probability $p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1})$ is the GP predictive distribution given in Eqs. (14)–(15). Computing the exact distribution $p(\boldsymbol{\Delta}_t)$ in Eq. (16) is analytically intractable. Therefore, we use exact moment matching to approximate this distribution by a Gaussian as detailed in [10].

The GP predictions at uncertain inputs allow us to predict the long-term outcome of a control strategy π for a distribution $p(\boldsymbol{x}_0)$ over start states, which results in the desired probability distribution over a trajectory $p(\boldsymbol{\tau}^\pi)$.

With the predictive distribution $p(\boldsymbol{\tau}^\pi)$ over trajectories, the formulation of model-based imitation learning as an RL problem with a natural cost function induced by the KL divergence, we introduced all ingredients for model-based imitation learning via probabilistic trajectory matching.

IV. Experimental Results

The performance of our imitation learning approach is demonstrated by a simulated double-pendulum swing-up task and a table-tennis hitting-task on a real compliant BioRob$^{\text{TM}}$ (shown in Fig. 1).

A. Learning to Swing up a Double Pendulum

The double pendulum consists of two actuated joints and two links. The states of the system were the joint positions and velocities $\boldsymbol{x} = [q_1, q_2, \dot{q}_1, \dot{q}_2]^\top$; the actions were the motor torques $\boldsymbol{u} = [u_1, u_2]^\top$. The robot was mounted fixed on the ground. The task was to swing-up and balance the two links of the double pendulum as visualized in Fig. 3.

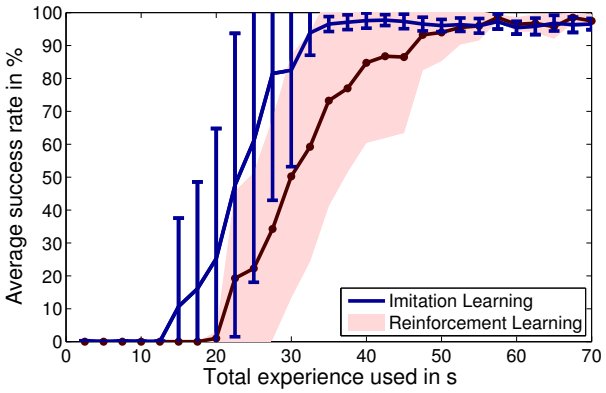


Fig. 4. Average success rate for solving the double-pendulum swing-up task. The horizontal axis shows required amount of data the algorithms effectively used for learning. The shaded red graph shows the success rate of an RL controller with a hand-crafted reward function[10]. The solid blue line shows the performance of our model-based imitation learning approach from Sec. III. Imitation learning converged significantly faster.

The motor torques at either of the joints were limited to the range $[-3, 3]\,\text{Nm}$. To explicitly take these torque limits into account during planning, we squashed a policy $\tilde{\pi}$ through a sinusoidal to obtain a torque-restricted policy

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}) = u_{\max} \sin(\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta})). \quad (17)$$

In this experiment, we used a nonlinear Radial Basis Function (RBF) network with axis-aligned Gaussian features ϕ to represent $\tilde{\pi}$. This policy can be written as

$$\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) \quad \text{with} \quad (18)$$

$$\phi_i(\boldsymbol{x}) = \exp\big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_i)^\top \boldsymbol{\Gamma}^{-1}(\boldsymbol{x} - \boldsymbol{c}_i)\big) \quad (19)$$

with weights w_i, centers $\boldsymbol{c}_i$, and length scales l_j, such that $\boldsymbol{\Lambda} = \text{diag}(l_1^2, l_2^2, \ldots, l_D^2)$. We learned the weights $\boldsymbol{w}$ and the Gaussian features $\boldsymbol{\phi}$ jointly. We used 100 basis functions ϕ in the RBF network. Therefore, the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^{816}$ were $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{\Lambda}, \boldsymbol{c}\}$.

We chose a sampling frequency of 10 Hz and a total prediction horizon of $T = 2.5\,\text{s}$. Five successful demonstrations of the task were generated to create a distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over expert trajectories according to Sec. II-A.

To qualitatively evaluate learning success, we compared our proposed imitation learning approach with an RL baseline. In particular, we used the data-efficient PILCO policy search method [10], also summarized in Algorithm 1. Note that PILCO required a hand-crafted cost function, whereas our imitation-learning approach matches trajectories by minimizing KL divergences.

The average success rate as a function of required data is visualized in Fig. 4. We defined success when the double pendulum performed the swing-up and balanced in the inverted position. The success rate is given in percent and averaged over 10 independent runs of the algorithms. The

shaded red graph represents PILCO's learning speed and reaches a success rate of $95\,\%$ after about $50\,\mathrm{s}$ of robot interactions. The performance of our imitation-learning algorithm with random initializations is visualized as the blue solid line and reaches a similar success rate after only $33\,\mathrm{s}$ only.

These results show that the KL divergence is an appropriate measure for matching trajectory distributions in the context of imitation learning. The performance boost over hand-crafted cost functions can be explained by the fact that RL with the hand-crafted cost functions initially needs to explore good trajectories leading to the desired configuration. Imitation learning instead has information about the desired trajectories through the expert's demonstrations, and, hence, does not rely on excessive exploration.

B. Learning a Ball Hitting Task with the BioRob™

We used the biomechanically-inspired compliant BioRob™ arm to learn a table-tennis hitting task from demonstrations, see Fig. 1. In the following, we describe first the robot hardware and experimental setup. After that we detail controller learning.

1) Hardware Description: We evaluate the proposed approach on a real robotic platform, the BioRob™ [16]. The BioRob™ is a compliant, light-weight robotic arm, capable of achieving high accelerations. Its design places the servo motors close to the torso, minimizing the inertia of the links and enabling the end-effector to move with high velocities. Experimental results have shown Cartesian velocities of the end-effector of up to $6.88\,\mathrm{m/s}$ [15]. Our BioRob X4, is equipped with an end-effector module that increases the total number of degree of freedom to 5. 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 robot's design has advantages over the traditional approaches, controlling a dynamic and 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, cable slacking effects, stiction, and the energy stored in the cable springs. As a result, the accuracy of both the forward and inverse dynamics models is insufficient to make the robot follow desired trajectories. Moreover, if the torques are not smooth, oscillations close to the eigen-frequency of the joints can occur. During the oscillations, the motors may hold the same position while the joints oscillate due to the kinematic decoupling from the motors. Active damping is non-trivial, as the control law has to incorporate both joint and motor positions, while considering the unmodeled nonlinearities.

2) Experimental Set-up: 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 Fig. 5. The shape of the racket alongside with the high velocities produced a significant amount of drag, which is generally hard to model accurately. Such undermodeling would lead to substantial errors in parametric models. Thus, first learning a data-driven non-parametric GP forward model, and, subsequently, learning control policies for solving the ball-hitting task, is particularly promising for this compliant system.

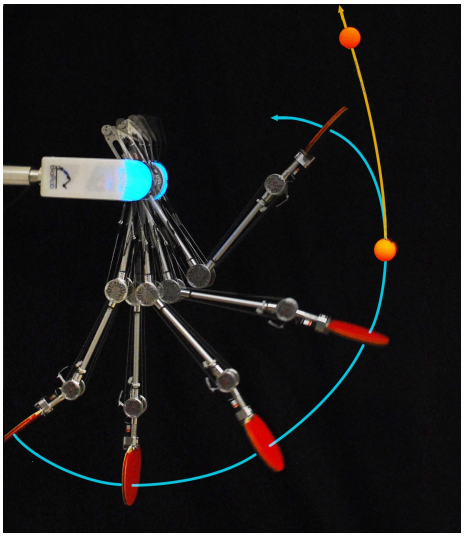
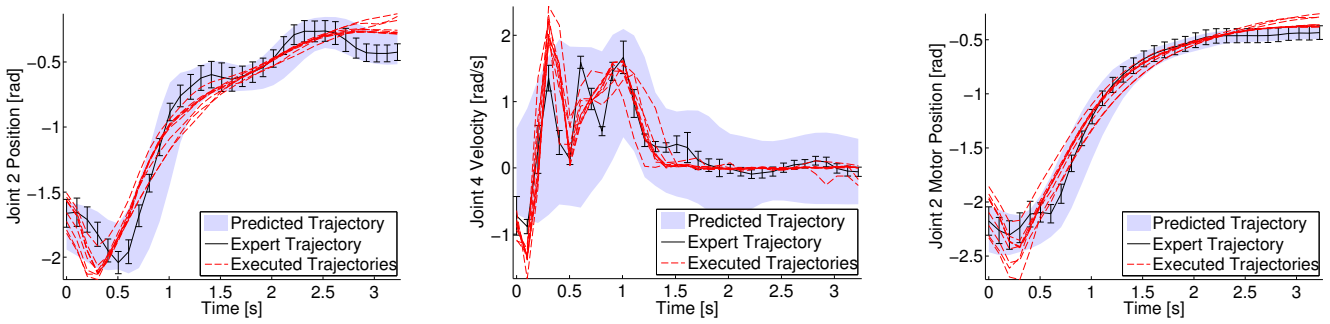


Fig. 5. Table-tennis hitting task using the compliant BioRob™. A trajectory of the ball hitting task is shown where the start configuration was the robot pointing downwards. The total time of the movement was $3.3\,\mathrm{s}$. The blue spline shows the trajectory of the table-tennis racket; the yellow spline shows the flight curve of the ball.

We used three joints of the BioRob™ for performing the ball-hitting task. The state $\boldsymbol{x} \in \mathbb{R}^9$ was given by three joint positions, velocities, and motor positions of the BioRob™; the actions $\boldsymbol{u} \in \mathbb{R}^3$ were the corresponding motor torques.

3) Controller Learning: To learn a BioRob™ controller for the ball-hitting task, we parametrized the policy by means of an RBF network as described in Eqs. (17)–(19). We chose a policy representation with 60 basis functions and a sampling frequency of $10\,\mathrm{Hz}$ for the forward model. The task of the robot was to hit a hanging ball with a racket as visualized in Fig. 5. For creating the distributions of the expert trajectory $p(\boldsymbol{\tau}^{\mathrm{exp}})$, we used three demonstrations via tele-operation of the task; we also used them for initializing the GP forward model. Our imitation-learning approach based on trajectory matching led to rapid learning. From the third trial onward, the BioRob™ reliably solved the ball-hitting task.

The predicted and demonstrated trajectory distributions along with some executed trajectories after 5 learning iterations of our imitation-learning approach are shown in Fig. 6. The figure shows the joint position, joint velocity, and motor position of different joints. The black error bars represent the distribution $p(\boldsymbol{\tau}^{\mathrm{exp}})$ of expert trajectories. The blue shaded graph is the trajectory prediction $p(\boldsymbol{\tau}^{\pi})$ of the GP model. The red dotted lines are executed trajectories of the controller from different starting positions. The executed rollouts of the robot were close to the predicted and expert trajectories. The robot robustly imitated the task with the learned controller from different starting positions, using a total of $\leq 40\,\mathrm{s}$ of



(a) Position of joint 2. (b) Velocity of joint 4. (c) Motor Position of joint 2.

Fig. 6. Learning results of the ball hitting task. The figure shows an example of (a) a joint position, (b) a joint velocity, and (c) a motor position of different joints of the BioRob^{TM}. The black error bar shows the distribution of the expert trajectory. The blue shaded graph is the distribution of the predicted trajectory. Both are plotted with twice the standard deviation. The red dotted lines are executed trajectories of the BioRob^{TM} with the learned controller. There start state was sampled from the initial distribution $p(\boldsymbol{x}_0)$.

data to learn both an accurate forward model and a good controller for imitating demonstrated trajectories.

V. Conclusion

In this paper, we have presented a novel approach to imitation learning that determines policies based on probabilistically matching predicted trajectories with demonstrated trajectories. Using the KL divergence for matching trajectories gives rise to a natural cost function that can be used by RL algorithms to learn the corresponding policies. In this paper, we used a sample-efficient policy search algorithm based on learned probabilistic forward models to determine the predicted trajectories. We have shown that our approach to imitation learning can substantially speed up learning. Moreover, we have demonstrated that our method is directly applicable to learning models and policies for a highly compliant robotic arm in only a few attempts.

References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In *Advances in Neural Information Processing Systems*. 2007.

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

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

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

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

[6] M. Bain and C. Sammut. A Framework for Behavioural Cloning. *Machine Intelligence*, 15:103–129, 1999.

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

[8] A. Billard, S. Calinon, and F. Guenter. Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks. *Robotics and Autonomous Systems*, 2006.

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

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

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

[12] K. Doya. Reinforcement Learning in Continuous Time and Space. *Neural Computation*, 12(1):219–245, 2000.

[13] W. A. Gruver, C. T. Thompson, S. D. Chawla, and L. A. Schmitt. CAD Off-line Programming for Robot Vision. *Robotics*, 1(2):77–87, 1985.

[14] J. Kober and J. Peters. Imitation and Reinforcement Learning. *IEEE Robotics & Automation Magazine*, 17(2):55–62, 2010.

[15] T. Lens. *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, 2012.

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

[17] C. Nehaniv and K. Dautenhahn. *Imitation in Animals and Artifacts*, chapter The Correspondence Problem. MIT Press, 2002.

[18] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous Inverted Helicopter Flight via Reinforcement Learning. In *International Symposium on Experimental Robotics*, 2004.

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

[20] J. Peters, K. Mülling, and Y. Altun. Relative Entropy Policy Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

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

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

[23] T. Raiko and M. Tornio. Variational Bayesian Learning of Nonlinear Hidden State-Space Models for Model Predictive Control. *Neurocomputing*, 2009.

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

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

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

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

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

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