Active Inverse Model Learning with Error and Reachable Set Estimates

Danny Driess* Syn Schmitt† Marc Toussaint*

***Abstract*— In this work, we propose a framework to learn an inverse model of redundant systems. We address three problems. By formalizing what it actually means to learn an inverse model, we derive a method where the inverse model, represented as a neural network, is learned by minimizing an upper bound on the real performance error, which is provided by a forward model (kernel regression or Gaussian process) learned on the currently available data. Most machine learning methods focus on learning the mapping of the function. For inverse models, it is, however, crucial to know the reachable set of the true forward model, since this becomes the domain of the inverse. Therefore, we secondly propose a method to estimate the reachable set of the system. Finally, we develop an active exploration strategy that is based on maximizing a lower bound on the true fill-distance to efficiently generate the data in the high dimensional input space. A key feature of our method is that the resulting learned inverse model provides error bounds on its performance.**

From an application point of view, this work is motivated by learning to control musculoskeletal systems. In the experiments, we show for both a simulated model of a human arm with six muscles and a real muscle-driven robot that the proposed method is able to learn the reachable set of these systems as well as a policy that enables to accurately control the position.

I. Introduction

Models play a central role in robotics and other disciplines. One can generally distinguish between forward models, which describe how a system reacts to control inputs, and inverse models, which predict the necessary control inputs that lead to a desired system change. While forward models are usually unique, in most interesting situations, there are infinitely many inverse models for the same system, prohibiting direct learning with standard regression techniques. Therefore, obtaining an inverse model, both from an analytical and learning point of view, is challenging.

Biomechanical musculoskeletal systems are one instance where this non-uniqueness of an inverse model occurs, since there are multiple, at least two, redundant muscles that drive one or even multiple joints. The complex, nonlinear dynamics, delays, hidden states etc. make it difficult to control muscle-driven systems with classical methods. The properties of the muscles and their redundant antagonistic setup lead to a certain intrinsic stability that is, in principle, favorable for learning. However, this redundant muscle setup is also the reason for the difficulties of learning an inverse for such systems.

The non-uniqueness problem of learning inverse models has extensively been studied in the context of inverse kinematics, e.g. [1], [2], [3], [4], [5], [6]. However, a crucial aspect missing in most existing approaches is to systematically and efficiently estimate the reachable set, i.e. to know which parts of the workspace can be reached at all. This is essential for several reasons: First, the inverse model is well-defined only over the reachable set. Therefore, when defining an objective for learning an inverse model, points outside the reachable set should be neglected, which requires to have an estimate of the reachable set. Secondly, in an active learning setting, the aim should be to cover the domain of the inverse model, which also requires to estimate the reachable set.

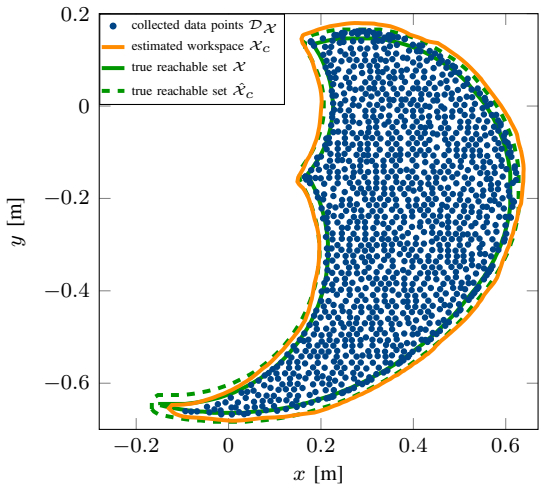


Fig. 1. Estimated reachable set and distribution of collected data points in the end-effector space after 1200 iterations for simulated human arm model.

Therefore, the problem of inverse model learning and reachable set estimation is inherently linked, should be treated jointly and, in an active learning setting, performed simultaneously. In the present work, we propose methods to this end, where bounds on the real performance error are the central piece to achieve all three aims:

First, by formalizing more rigorously what it actually means to learn an inverse model, we derive a method where the inverse model, represented as a neural network, is learned by minimizing an upper bound on the real performance error. This upper bound is provided by a forward model (kernel or Gaussian process regression), which is trained on the currently available data.

We secondly propose a method to estimate the reachable set based on the derived error bound of the current learned forward and inverse model. This estimated reachable set is also efficient to compute.

Thirdly, we derive an active exploration strategy to efficiently generate the training set of the forward model with the goal of thereby improving the inverse model. This is realized by maximizing a lower bound on the true fill-distance of the real workspace (the domain of the inverse model),

* Machine Learning and Robotics Lab, University of Stuttgart, Germany.
† Biomechanics and Biorobotics Group, University of Stuttgart, Germany.
This work was funded by the Baden-Württemberg Stiftung.

which is again provided through the error estimate. This strategy inherently trades-off exploration and exploitation and explores in the low dimensional workspace instead of the high dimensional control input space.

A key feature of the proposed framework is that the learned inverse model provides guaranteed upper bounds on its performance when applied to the real system. These bounds are easy to compute and, as we will show in the experiments, are also suitable in practice.
To summarize, our main contributions are

- Formalization of inverse model learning
- Estimation of reachable set
- Active exploration strategy
- Estimate (upper bound) of the real performance error of the learned inverse model

The rest of the paper is organized as follows. After reviewing related work in Sec. II, the background about the function approximator for the forward model is given in Sec. III, where we also state an error estimate. The main methodology is developed in Sec. IV. Experimental results with a simulated human arm and a real muscle-driven robot as well as an ablation study are presented in Sec. VI.

II. Related Work

A. Learning Inverse Models

Many methods address the non-uniqueness of inverse models specifically in the context of inverse kinematics. Simple regression methods are not applicable, since averaging over multiple configurations that correspond to the same task value leads to invalid solutions [1], [5]. One way to handle this averaging is to weight the data samples according to some objective [1], [3], [7]. For example, the authors of [1] consider those samples as more important that are closer to a homing position of the robot. In [8], it was shown that such weightings do not perform well for musculoskeletal systems, since the choice of a suitable weighting objective is unclear.

Another method to resolve the non-uniqueness is to first learn a forward model, for which standard regression techniques can be used [5]. The inverse itself is then obtained as a right inverse to the learned forward model. This way, the redundancy is resolved implicitly through the learned forward model. While [5] considers one global inverse model, another approach is to learn multiple paired forward and inverse models [9], [6], [4]. The main idea of those approaches is that each forward-inverse pair represents different, smaller parts of the space. Depending on the current context, which could be the current state for example, the responsibility of the different inverse models is chosen. One disadvantage of most of these methods is that the learning of the inverse model does not take into account that the learned forward model is imperfect. If the data for the forward model is sparse, as we show in the experiments, this could lead to undesired results. Speaking of data, most works also consider mainly given datasets or specify the data generation manually.

Generating the data efficiently is, however, an important problem for the success of an inverse learning method. Due to the high dimensionality of the control input space, dense sampling is prohibitive. To address this issue, several authors propose to explore in the workspace/goal space instead of the control input space [3], [10], [11], [8]. By bootstrapping the iteratively learned inverse model, so-called goal babbling approaches [3], [10], [8] showed to generate the data to obtain an inverse model sample efficiently in high dimensional control input spaces. A limitation of those goal babbling methods is that either the exploration targets in the goal space are manually specified or chosen heuristically by knowing the true workspace. A different approach to generating targets for the exploration in the goal space is intrinsic motivation, for example by estimating the competence progress [11], which also shows impressive results for challenging tasks. These methods, however, also do not systematically estimate the reachable set. The only work we are aware of that deals with estimating the reachable set for learning inverse models is [12]. However, there is no representation of the reachable workspace which could be computed easily and the exploration in one direction stops if a heuristically chosen criterion indicates the end of the workspace.

The present work is, to our knowledge, the first to systematically approach the estimation of the reachable set from learned models in form of a representation that is also computationally feasible to determine.

B. Active Learning in Robotics

Active learning deals with generating data in a most informative manner. For example, Bayesian optimization methods have successfully been applied to tuning controller parameters for robotic applications [13], [14]. Most active learning methods are derived under the assumption to be able to sample the unknown function at a chosen location. In the case of inverse model learning, when exploring in the workspace to circumvent the high dimensionality of the input space, it is not possible to query an arbitrary desired location in the workspace, since this would require an already known perfect inverse. In the present work, we therefore derive an active learning principle for exploration in the workspace that takes the imperfection of the inverse model into account.

C. Learning to Control Musculoskeletal Systems/Robots

Several authors consider learning controllers for pneumatically driven robots [15], [16], [17], [8], [18], [19], [17]. In [15] and [16] the redundancy problem is circumvented since only a control for one of the antagonistic muscles is learned.

In [8] and [18] the non-uniqueness is resolved by only learning a forward model. The actual inverse query is then obtained by solving a non-convex optimization problem. Both [8] and [18] include a term in the optimization objective that ensures that the optimization problem stays close to the collected data. In the present work, we derive a similar term more rigorously. With respect to the data generation, in [8] the targets in the workspace are specified manually, while in [18] a preexisting controller is needed to generate the data.

III. Kernel Regression

An important part of this work is to learn a function with the ability, under some assumptions, to estimate the error

between the true and the learned function. Kernel based methods enable to derive such error estimates elegantly. In the following, we provide background about kernel ridge regression and derive an error bound in the regularized case, which we have not seen in this form in the literature yet, although it is a straightforward extension to known results.

A. Kernel Ridge Regression

Let $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a positive definite kernel for the nonempty set $\mathcal{U} \subset \mathbb{R}^m$. Denote H_k with norm $\|\cdot\|_{H_k}$ and inner product $\langle \cdot, \cdot \rangle_{H_k}$ the reproducing kernel Hilbert space (RKHS) associated with k. Assume that data is generated by a function $\hat{\phi} : \mathcal{U} \to \mathbb{R}$, i.e. $\mathcal{D}_1 = \{(\mathbf{u}_i, x_i)\}_{i=1}^n$, $\mathbf{u}_i \in \mathcal{U}$, $x_i \in \mathbb{R}$, where $x_i = \hat{\phi}(\mathbf{u}_i)$. Kernel ridge regression with the regularization parameter $\sigma \geq 0$ is then defined by

$$\min_{\phi \in H_k} \sum_{i=1}^{n} |x_i - \phi(\mathbf{u}_i)|^2 + \sigma^2 \|\phi\|_{H_k}^2 \tag{1}$$

with the unique solution

$$\phi(\mathbf{u}) = \mathbf{x}^T \mathbf{G}^{-1} \boldsymbol{\kappa}(\mathbf{u}), \tag{2}$$

where $\mathbf{G} = \mathbf{K} + \sigma^2 \mathbf{I} \in \mathbb{R}^{n \times n}$, $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^{n \times n}$, $\mathbf{x} = (x_i)_{i=1}^n \in \mathbb{R}^n$, $\boldsymbol{\kappa}(\mathbf{u}) = (k(\mathbf{u}_i, \mathbf{u}))_{i=1}^n \in \mathbb{R}^n$. Readers familiar with Gaussian process regression see the similarity to the mean function.

B. Error Estimate

Assume that the true function $\hat{\phi} : \mathcal{U} \to \mathbb{R}$ fulfills $\hat{\phi} \in H_k$ and $\|\hat{\phi}\|_{H_k} < \infty$, then for all $\mathbf{u} \in \mathcal{U}$ it holds

$$\left| \hat{\phi}(\mathbf{u}) - \phi(\mathbf{u}) \right| \leq \left\| \hat{\phi} \right\|_{H_k} s(\mathbf{u}), \tag{3}$$

where

$$s(\mathbf{u}) = \sqrt{k(\mathbf{u}, \mathbf{u}) - \boldsymbol{\kappa}(\mathbf{u})^T \mathbf{G}^{-1} \boldsymbol{\kappa}(\mathbf{u}) - \sigma^2 \boldsymbol{\kappa}(\mathbf{u})^T \mathbf{G}^{-2} \boldsymbol{\kappa}(\mathbf{u})}. \tag{4}$$

The proof, which follows a similar idea as in [20], where the case of $\sigma = 0$ is considered, has to be omitted due to space constraints. Note that since $\mathbf{G}^{-2}$ is positive definite, this bound is tighter than known results that are based on the variance estimate of the Gaussian process.

IV. Active Inverse Model Learning

In this section, we first precisely state what we consider as learning an inverse model. We then identify the involved challenges, for which we provide solutions.

A. Problem Definition

Let $\mathcal{U} \subset \mathbb{R}^m$ be the set of control inputs, e.g. muscle stimulations as considered here in this work, and $\mathcal{X} \subset \mathbb{R}^d$ be the so-called reachable set or the workspace. Further, assume that $\mathcal{U}$ is compact and that $d \leq m$, which corresponds to redundancy. The *true* system

$$\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X} \tag{5}$$

is a function that statically and uniquely maps a control input $\mathbf{u} \in \mathcal{U}$ to a point $\mathbf{x} \in \mathcal{X}$ in the reachable set. In the context of musculoskeletal systems as discussed in Sec. V, $\widehat{\boldsymbol{\phi}}$ describes the equilibrium configuration $\mathbf{x}$ that is reached for a vector of static muscle stimulations $\mathbf{u}$.

The goal of this work is to learn both the reachable set $\mathcal{X}$ as well as an inverse model

$$\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U} \tag{6}$$

such that

$$\forall_{\mathbf{x} \in \mathcal{X}} \ : \ \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) = \mathbf{x}. \tag{7}$$

By setting $\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U})$, such a (right) inverse automatically exists, but due to the redundancy, there are typically infinitely many of them. To represent the inverse model, we chose a standard feedforward neural network with continuous activation functions and an output layer that ensures that $\boldsymbol{\pi}$ maps into $\mathcal{U}$. Therefore, $\boldsymbol{\pi} \in C(\mathcal{X}, \mathcal{U})$. If we further assume that the true forward model is continuous as well and $\mathcal{X}$ does not contain isolated points, condition (7) is equivalent to

$$\int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x} = 0. \tag{8}$$

However, inverses are in general not continuous, which means that (8) cannot be fulfilled exactly for a continuous class of $\boldsymbol{\pi}$. Therefore, ideally, the parameters $\mathbf{w}$ of the inverse model, i.e. the weights of the neural network, would be chosen to optimize

$$\min_{\mathbf{w} \in \mathcal{W}} \int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x}. \tag{9}$$

Unfortunately, we neither assume to know $\mathcal{X}$, nor $\widehat{\boldsymbol{\phi}}$ is available in terms of closed form expressions. The only way to acquire knowledge about the system is to evaluate $\widehat{\boldsymbol{\phi}}(\mathbf{u})$ for a specific control input. Therefore, (9) is of little use. To overcome this, the first idea would be to replace the true forward model with a learned surrogate $\boldsymbol{\phi}$. If the dataset is rich enough, one could set $\mathcal{X} \approx \boldsymbol{\phi}(\mathcal{U})$ and replace $\widehat{\boldsymbol{\phi}}$ in (9) with the learned $\boldsymbol{\phi}$. However, evaluating $\widehat{\boldsymbol{\phi}}(\mathbf{u})$ could involve a costly numerical simulation or even a real robot experiment, which, together with the high dimensionality of the control input space, prohibits dense sampling in $\mathcal{U}$ to build a rich enough dataset for learning $\boldsymbol{\phi}$ everywhere in $\mathcal{U}$. Since the optimization problem (9) treats the forward model as exact, especially if $\boldsymbol{\phi}$ is learned based on little data, an unreasonable inverse could be learned. Indeed, as we will empirically show in the experiments in Sec. VI-B.1, simply using (9) with an iteratively learned $\boldsymbol{\phi}$ is not sufficient and leads to bad performance for a learned $\boldsymbol{\phi}$ from little data.

To summarize, learning an inverse model requires

- A surrogate objective to formulate the inverse learning problem in a way that takes into account that the forward model is learned from little data.
- An exploration strategy to efficiently generate the data in the high dimensional input space to learn the forward model with the aim to reduce the inverse model's error.
- A way to estimate the reachable set $\mathcal{X}$.

B. Learning the Inverse Model $\boldsymbol{\pi}$

During the learning process, the dataset $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^n$, consisting of control inputs $\mathbf{u}_i \in \mathcal{U}$ that lead to the configuration $\mathbf{x}_i \in \mathcal{X}$, is collected. Based on this data, the forward model $\boldsymbol{\phi} : \mathcal{U} \to \mathcal{X}$, $\boldsymbol{\phi}(\mathbf{u}) = (\phi_1(\mathbf{u}), \dots, \phi_d(\mathbf{u}))^T$ is learned, where each component ϕ_i is a (separate) solution of the kernel ridge regression problem (1) with (own) reproducing kernel Hilbert space H_{k_i}. Given this learned forward model, the true reaching error at $\mathbf{x} \in \mathcal{X}$ for an inverse model $\boldsymbol{\pi}$ can be estimated as

$$\begin{aligned}\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}\right\| &= \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))+\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}\right\| \\ &\leq \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}\| + \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))\right\|. \end{aligned} \tag{10}$$

The first term describes how well the inverse model is a right inverse to the learned forward model. The second term is the error between the true and the learned forward model at $\boldsymbol{\pi}(\mathbf{x})$. Under the assumption that each component $\hat{\phi}_i$ of the true forward model $\widehat{\boldsymbol{\phi}}$ has a bounded RKHS norm in H_{k_i}, using the kernel ridge regression error estimate (3) for each pair $\hat{\phi}_i, \phi_i$, the true reaching error in the $\mathcal{X}$ space for an arbitrary inverse $\boldsymbol{\pi}$ can be bound by

$$\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| \leq \varepsilon(\mathbf{x}), \tag{11}$$

where we define our error estimate as

$$\varepsilon(\mathbf{x}) = \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\| + \sqrt{\sum_{i=1}^d \left\|\hat{\phi}_i\right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2}. \tag{12}$$

Therefore, we have derived an upper bound on the quality of the inverse model

$$\int_{\mathcal{X}} \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\|^2 \mathrm{d}\mathbf{x} \leq \int_{\mathcal{X}} \varepsilon(\mathbf{x})^2 \,\mathrm{d}\mathbf{x} \tag{13}$$

$$\leq 2\int_{\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\|^2 + \sum_{i=1}^d \left\|\hat{\phi}_i\right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2 \,\mathrm{d}\mathbf{x} \tag{14}$$

$$\leq 2\int_{\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\|^2 + \sum_{i=1}^d \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2 \,\mathrm{d}\mathbf{x} \tag{15}$$

that does not require the true forward model, except for an upper bound $\left\|\hat{\phi}_i\right\|_{H_{k_i}} \leq \beta_i$ on its RKHS norm.

As discussed before, there are potentially infinitely many inverse models. To have control over which inverse would be learned, we introduce a positive, monotone function $l : \mathcal{U} \to \mathbb{R}^+$, for example $l(\mathbf{u}) = \|\mathbf{u}\|^2$ would encourage to learn inverses with low control inputs.

Plugging everything together, the inverse model is learned by the optimization problem

$$\begin{aligned}\min_{\mathbf{w}\in\mathcal{W}} \int_{\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^d \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))^2 \\ + \eta\, l(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))\,\mathrm{d}\mathbf{x}. \end{aligned} \tag{16}$$

$\eta \geq 0$ is a trade-off parameter. Intuitively, the optimization problem tries to find an inverse to the learned forward model, while, which is expressed with the second term in this objective, staying close to the data.

C. Estimating the Workspace $\mathcal{X}$

The integral formulation (16) assumes to know the true workspace $\mathcal{X}$. While in some situations one might know $\mathcal{X}$ a priori, in general, we would like to avoid this. Here, we propose one way to estimate $\mathcal{X}$ based on the current learned forward and inverse model, utilizing the error estimate.

It is clear that one cannot expect to estimate the reachable set exactly. Instead, we define for an error certainty $c > 0$

$$\hat{\mathcal{X}}_c = \left\{\mathbf{x} \in \mathbb{R}^d : \mathrm{dist}(\mathbf{x}, \mathcal{X}) < c\right\} \tag{17}$$

as the true reachable set expanded by c. With this definition, we consider as learning the reachable set as estimating $\hat{\mathcal{X}}_c$. Since $\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) \in \mathcal{X}$ for every $\mathbf{x} \in \mathbb{R}^d$ (where $\boldsymbol{\pi}$ can be evaluated, which is $\mathbb{R}^d$ for a neural network), we have

$$\mathrm{dist}(\mathbf{x}, \mathcal{X}) = \inf_{\mathbf{x}'\in\mathcal{X}} \|\mathbf{x}' - \mathbf{x}\| \leq \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| \leq \varepsilon(\mathbf{x}), \tag{18}$$

where we use our derived error estimate (12) of the inverse model. Therefore, we propose to estimate the reachable set with error certainty $c > 0$ as

$$\mathcal{X}_c = \left\{\mathbf{x} \in \mathbb{R}^d : \varepsilon(\mathbf{x}) < c\right\}, \tag{19}$$

which, due to (18), automatically has the property

$$\mathcal{X}_c \subset \hat{\mathcal{X}}_c, \tag{20}$$

meaning that $\mathcal{X}_c$ never overestimates the true $\hat{\mathcal{X}}_c$. Since $\mathcal{X}$ is assumed to be low dimensional, calculating $\mathcal{X}_c$ is also feasible, as compared to evaluating $\boldsymbol{\phi}(\mathcal{U})$ or even $\widehat{\boldsymbol{\phi}}(\mathcal{U})$.

D. Exploration Strategy

As a final step, we discuss how the data for learning the forward model can be collected in a sample efficient way. The exploration strategy should select points such that they become dense in the whole unknown workspace $\mathcal{X}$. This can be expressed in terms of the so-called fill-distance

$$\max_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\|, \tag{21}$$

where $\mathcal{D}_\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ is the $\mathbf{x}$-part of the dataset $\mathcal{D}$. Intuitively, the fill-distance denotes the radius of the largest ball with center in $\mathcal{X}$ that does not contain any already sampled datapoint from $\mathcal{D}_\mathcal{X}$. In our case, we define the true fill-distance as

$$\max_{\mathbf{u}\in\mathcal{U}} \left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \left\|\widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i\right\| \right), \tag{22}$$

which measures how large unexplored parts in the true $\mathcal{X}$ are. Similar to the discussion in the last paragraphs, in contrast to many other active learning methodologies, where one can just query the unknown function at a desired location, we cannot query a $\mathbf{u}$ such that $\widehat{\boldsymbol{\phi}}(\mathbf{u}) = \mathbf{x}^*$, where $\mathbf{x}^*$ would maximize the fill-distance, since this would require an already perfectly known inverse. Therefore, the goal of this section is to derive a computationally feasible lower bound on the true fill distance.

Based on the simple calculation

$$\|\mathbf{x}-\mathbf{x}_i\| = \left\|\mathbf{x}-\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))+\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}_i\right\| \tag{23}$$

$$\leq \left\|\mathbf{x}-\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))\right\| + \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}_i\right\| \tag{24}$$

$$\leq \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}_i\right\| + \varepsilon(\mathbf{x}), \tag{25}$$

where we have used our error estimate (12), a lower bound on the true fill distance is

$$\sup_{\mathbf{x}\in\mathbb{R}^d}\left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}}\|\mathbf{x}-\mathbf{x}_i\|-\varepsilon(\mathbf{x})\right) \tag{26}$$

$$\leq \sup_{\mathbf{x}\in\mathbb{R}^d}\left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}}\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}))-\mathbf{x}_i\right\|\right) \tag{27}$$

$$\leq \max_{\mathbf{u}\in\mathcal{U}}\left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}}\left\|\widehat{\boldsymbol{\phi}}(\mathbf{u})-\mathbf{x}_i\right\|\right), \tag{28}$$

since $\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbb{R}^d)) \subseteq \widehat{\boldsymbol{\phi}}(\mathcal{U})$. Let $\mathbb{X}$ be compact with $\mathcal{X} \subseteq \mathbb{X} \subset \mathbb{R}^d$. Then, in each iteration, the next target is chosen as

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}\in\mathbb{X}}\left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}}\|\mathbf{x}-\mathbf{x}_i\|-\varepsilon(\mathbf{x})\right) \tag{29}$$

and $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x}^*)$ is applied to the system. This exploration strategy therefore chooses exploration targets in the low dimensional $\mathcal{X}$-space instead of the high dimensional control input space and utilizes the current learned inverse model to generate the data for the forward model. Note we assumed here that $\boldsymbol{\pi}$ can be queried on $\mathbb{R}^d \supset \mathbb{X} \supseteq \mathcal{X}$, which for usual neural networks is always fulfilled.

E. Practical Remarks

In practice, calculating (16) with the estimated workspace is computationally infeasible, since the integral over $\mathcal{X}_c$ cannot be expressed in closed form. However, since we assume that the workspace is low-dimensional, we can evaluate ε on a d-dimensional grid. This gives a discretized version $\tilde{\mathcal{X}}_c$ of $\mathcal{X}_c$. Especially in the beginning of the exploration, $\mathcal{X}_c$ is often empty. Therefore, we include the already sampled data points $\mathcal{D}_\mathcal{X}$ to the estimated workspace. This leads to the optimization problem for the inverse model

$$\min_{\mathbf{w}\in\mathcal{W}}\sum_{\mathbf{x}\in\tilde{\mathcal{X}}_c\cup\mathcal{D}_\mathcal{X}}\Bigg(\|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))-\mathbf{x}\|^2+\sum_{i=1}^{d}\beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))^2 + \eta\, l(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))\Bigg). \tag{30}$$

In the exploration strategy objective (29), the new point is found by optimizing over a sufficiently large compact $\mathbb{X} \supseteq \mathcal{X}$ to guarantee that (29) is well-defined. While the derived lower bound (28) on the true fill-distance under the made assumptions even holds on $\mathbb{R}^d$, we found that the robustness of the methodology in practice with imperfect hyperparameters is increased if we constrain the exploration optimization problem to search inside a trust region around the already sampled data points, i.e. only those points that have a maximum distance of γ from the data points are considered in (29). Note that in most of our experiments, it would still work without this trust region, but not as reliably.

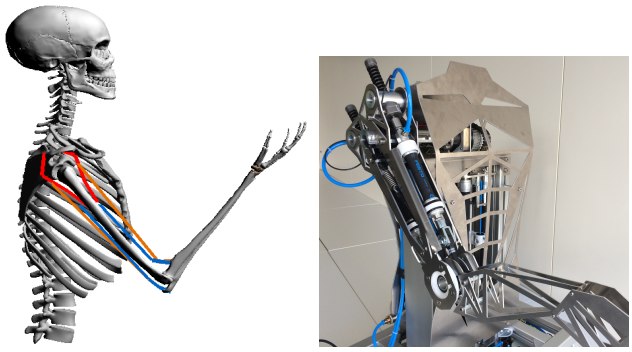


Fig. 2. Left: Simulation model of human arm with elbow and shoulder joint. Red lines are the two monoarticular shoulder muscle tendon units (MTUs), orange the two biarticular ones and blue depicts the two monoarticular elbow MTUs. Right: Real muscle-driven robot with two joints, articulated by five pneumatic muscle spring units (MSUs).

To solve the max-min optimization problem of the exploration strategy, we first sort $\mathcal{D}_\mathcal{X}$ into a nearest-neighbor tree. Then (29) can be solved easily by grid evaluation. Compared to the time required to run the simulations, i.e. querying $\widehat{\boldsymbol{\phi}}$, and training the neural network, the computation time for solving the exploration strategy optimization problem by this method was neglectable in our experiments.

V. Biomechanical Model of a Human Arm

We briefly describe the used biomechanical models. For details, refer to [8], where these models were introduced.

A biological movement apparatus is articulated by so-called muscle-tendon-units (MTUs). Since muscles can only actively contract, at least two muscles for each joint in an antagonistic setup are required. As we have discussed in [8], this redundant antagonistic setup, together with the characteristic nonlinear force-length and force-velocity relations of the MTUs, have the consequence that a vector of constant muscle stimulations $\mathbf{u} \in \mathcal{U}$ results in a specific equilibrium configuration, where small perturbations are counteracted by the passive visco-elastic properties of the MTU without the need of control [8]. The true forward model $\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X}$ for such a system therefore describes the equilibrium configuration $\mathbf{x} \in \mathcal{X}$, e.g. the position of the arm, when applying the constant muscle stimulation $\mathbf{u} \in \mathcal{U}$. The muscle stimulation space is normalized, i.e. $\mathcal{U} = [0,1]^m$. By using a sigmoid output layer for the inverse model, $\boldsymbol{\pi}(\mathbb{R}^d) \subset \mathcal{U}$ is automatically fulfilled.

A. Simulation Model

The simulation model (Fig. 2 left) consists of the upper and lower arm, connected by the elbow and shoulder joint, which are driven by $m = 6$ MTUs. For each joint, there are two monoarticular MTUs. In addition, two biarticular MTUs drive both joints. The MTU forces are calculated based on an extended Hill-type muscle model from [21].

B. Real Bio-Inspired Robot

Our developed real bio-inspired robot arm (Fig. 2 right) mimics the biological archetype. The two joints are actuated by five pneumatic muscles. Together with added serial springs, these muscle-spring-units (MSUs) have similar active and passive characteristics as MTUs. There are two

monoarticular MSUs for each joint and one biarticular MSU. The control input $\mathbf{u} \in [0,1]^5$ corresponds to pressures of 0 to 5 bar. Compared to the simulated arm model, the possible motion range is much smaller, since the pneumatic muscles have a limited stretch and contract capability.

VI. Experiments

We show both in simulation and with a real robot that the proposed method is able to accurately learn an inverse model and its reachable set. A video attachment is also available.

Each data point is generated by applying the constant muscle stimulation predicted by the current learned inverse model for the target specified by the proposed exploration strategy. The first 22 data points in simulation and 50 for the real robot were sampled with noise according to [8]. The neural network for $\boldsymbol{\pi}$ consists of 2 hidden layers with 300 units and relu activation functions. After a data point is added, the network is trained to optimize (30) for 30 episodes with stochastic gradient descent and a learning rate of 0.0005. We used a Gaussian kernel for the forward model with length-scale of 0.2. The bound on the RKHS norm of the true forward model was $\beta_i = 2$. Other hyperparameters were $\sigma = 0.001$, $\eta = 10^{-5}$, $c = 0.02$, $\gamma = 0.1$.

A. Simulated Human Arm Model

Fig. 3 shows the evolution of the learned reachable set as well as the distribution of the collected data points in the $\mathcal{X}$-space. The error certainty value for the estimated reachable set was $c = 0.02$, meaning that inside $\mathcal{X}_c$, i.e. inside the orange set in Fig. 1, 3, the inverse model estimates its real reaching error to be smaller than 2 cm. In the beginning of the procedure, the exploration strategy focuses on expanding the reachable set. After only 150 data points (Fig. 3d), the reachable set is nearly completely covered. Then, the exploration strategy automatically focuses on improving the model inside the already estimated reachable set. In Fig. 1, the final result after 1200 collected points is visualized. One can see that the estimated reachable set $\mathcal{X}_c$ matches the true one $\hat{\mathcal{X}}_c$ closely and the data points are relatively uniformly distributed, indicating that the exploration strategy is able to closely match the true fill-distance.

In order to test the accuracy of the learned inverse model at various stages of the exploration progress, after 100 collected data points each, the point reaching performance is tested on 137 target points inside $\mathcal{X}$. These target points and the actual reached points after 1200 collected data points are shown in Fig. 4a. The evolution of the actual error and its estimate over the number of collected data points is shown in Fig. 4b. One can see that in the beginning the error estimate is a bit more conservative, but in the end it is a tight upper bound. The final reached mean error is 1.50 ± 0.42 mm, the error estimate predicts 2.07 ± 0.40 mm. The reported values are the mean errors and standard deviations of 5 experiments started from scratch with the same parameters but different random seeds. In Fig. 4c, the trajectories of an out-of-center reaching experiment is shown after only 150 collected data points. As one can see, the actual reached positions are within the red circles, which indicate the error estimate $\varepsilon(\mathbf{x})$.

One cannot expect that the true forward model, i.e. the musculoskeletal system, comes from the same RKHS as the chosen kernel with its hyperparameters. Therefore, $\mathcal{X}_c$ sometimes overestimates $\hat{\mathcal{X}}_c$ slightly. However, as seen both by the estimated reachable set and the error estimate for the point reaching evaluation, the assumptions are also in practice reasonably well fulfilled, which make the error bounds usable in practice, without the need for extensive parameter tuning of β_i. Note that the true reachable set is also only an estimation, which neglects that the joint limits are modeled in the simulator as soft constraints. Therefore, there are some sampled points which are outside of $\mathcal{X}$.

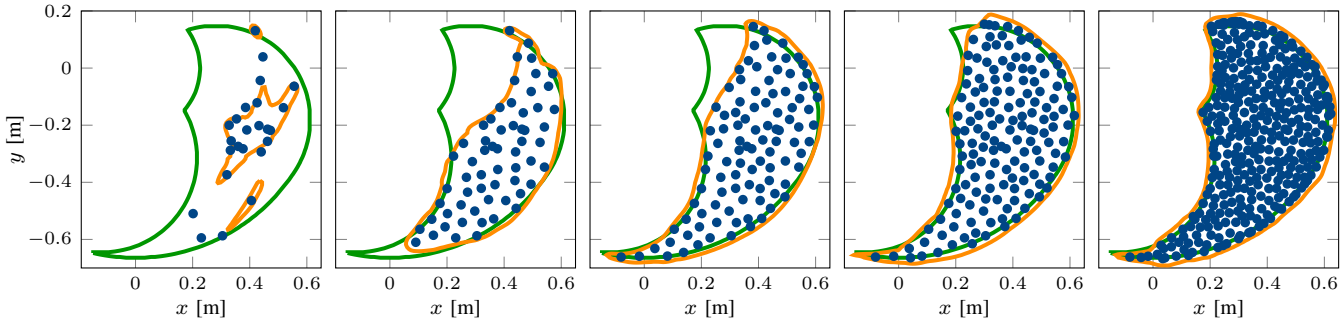
B. Ablation Study

In this experiment, we investigate the importance and influence of the various parts of the proposed framework on the point reaching performance. The point reaching targets are the same as in Fig. 4a. Except for the part that is changed, the rest of the methodology stays the same. All experiments are repeated 5 times from scratch with same parameters, but different random seeds.

1) Neglecting the Upper Bound: Here we consider what happens if the inverse model is not learned by optimizing the upper bound on the real performance, but only as an inverse to the learned forward model. Technically, one can think of setting $\beta_i = 0$ in (16), which corresponds to treating the learned forward model as perfect everywhere in $\mathcal{U}$. All other parts of the framework, i.e. the exploration strategy and the reachable set estimate for the integral stay the same and still use the error estimate. As one can see in Fig. 5a, where the resulting point reaching performance is shown for two different learning rates of the neural network, there is nearly no learning progress at all and the error is very high. The main reason for this is that there are large parts of the reachable set which remain unexplored, since, when neglecting the upper bound, the inverse model does not have good extrapolation and hence exploration capabilities. But also in explored parts of the workspace, the error is considerably higher than if the upper bound is included.

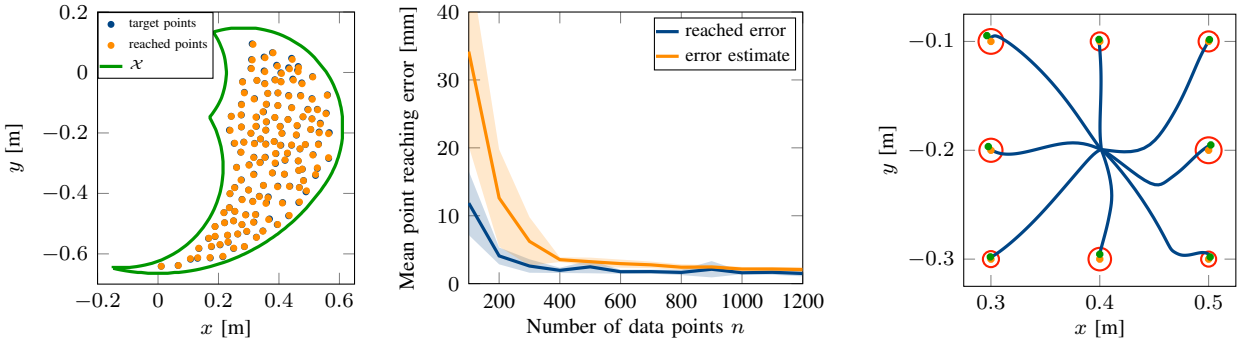
2) Influence of Integral: Next we investigate the influence of the value of c for estimating the reachable set for the integral part in the objective as well as the case if there is no integral at all and the inverse model is learned on the observed data only. Fig. 5b visualizes the reached performance error. As one can see, if c is decreased from 0.02 to 0.01, the error in the beginning is higher. The (empirical) reason for this is that if c is larger, the model extrapolates better. After about 500 data points, there is no statistically relevant difference, since then mainly the exploration focuses on the inside of $\mathcal{X}_c$. However, if c is made too large (in this case 0.05), the final reached error increases. This is caused by the fact that for large c, $\mathcal{X}_c$ contains points that are too far away from $\mathcal{X}$ and hence irritate the inverse learning. In case the integral is neglected completely, the performance significantly drops (about 3 times higher error).

3) Exploration Strategy Comparison: In Fig. 5c, different exploration strategies, with and without the integral, are compared to the proposed method. Random sampling in $\boxed{\mathcal{X}}$



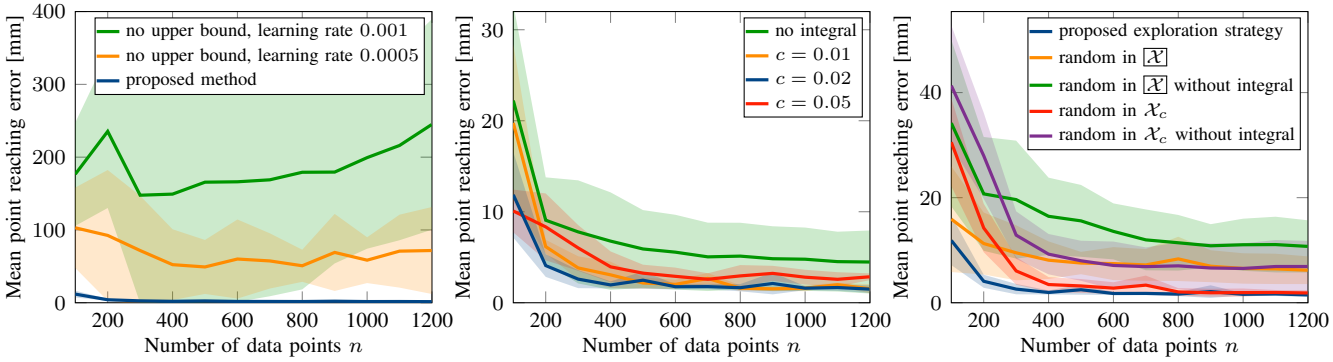
(a) $n = 24$ (b) $n = 60$ (c) $n = 100$ (d) $n = 150$ (e) $n = 350$

Fig. 3. Exploration evolution for a growing number n of collected data points with simulated human arm. Orange lines denote the boundary of the estimated reachable set, green lines the boundary of true reachable set. Blue points are the reached hand positions



(a) Point reaching, $n = 1200$ (b) Point reaching error and its estimate (c) Out-of-center reaching after 150 data points

Fig. 4. (a) point reaching evaluation after 1200 collected data points. (b) evolution of the reaching error and its estimate at different stages of the exploration. The red circles in (c) denote the error estimate, orange the targets, green the actual reached points and blue the trajectories



(a) Neglecting the upper bound (b) Influence of the integral over $\mathcal{X}_c$ (c) Exploration strategy comparison

Fig. 5. Ablation study: Influence of the various parts of the proposed method on the point reaching error

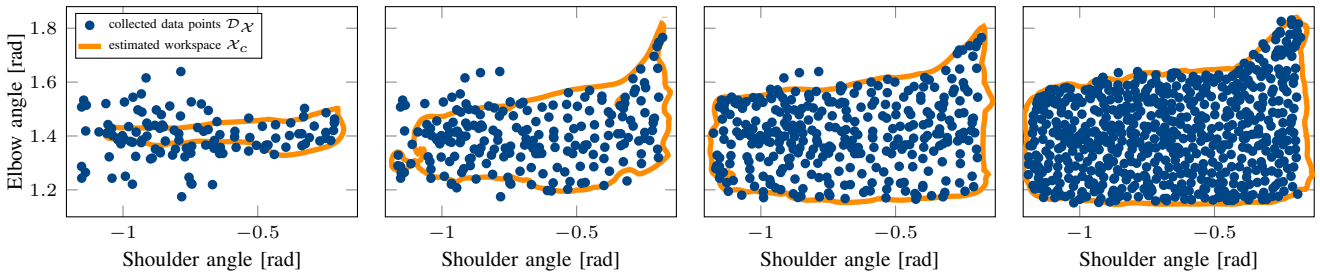
(orange and green) means uniform sampling in a sufficiently large bounding box of $\mathcal{X}$. Random in $\mathcal{X}_c$ means uniform sampling in the current estimated reachable set. The simple random sampling method in $\boxed{\mathcal{X}}$ has an about 4.1 times higher error than the proposed method. This is due to the fact that many targets are chosen outside the true reachable set, which means that those points do not contribute to increasing the performance of the inverse model.

With a final reached accuracy of 1.86 ± 0.54 mm, random sampling inside the estimated reachable set is competitive in the end to the proposed strategy. One has to note that through the estimated reachable set a similar effect as the lower bound on the real fill-distance is achieved, since both are derived based on the error estimate. The proposed exploration strategy is, however, in the beginning faster than random sampling in $\mathcal{X}_c$.

For both random sampling methods, the performance also significantly drops if no integral is considered. This especially holds true for random sampling in $\boxed{\mathcal{X}}$ (7.1 times higher error), since the integral helps for compensating areas inside $\mathcal{X}_c$ where no data has been observed.

C. Real Muscle-Driven Robot

With the real robot, the goal of the inverse model is to find pressure values to reach desired joint configurations. Therefore, the $\mathcal{X}$-space is the joint space of the robot. All parameters are the same as in the simulated experiment, except for the bound on the RKHS norm of the true forward model specifically for the exploration strategy, which was decreased to $\beta_i = 1$. Fig. 6 shows the evolution of the learned reachable set as well as the distribution of the collected data points. Looking at the shape of the estimated reachable set



(a) $n = 120$ (b) $n = 210$ (c) $n = 330$ (d) $n = 651$

Fig. 6. Exploration evolution with real robot for a growing number n of collected data points

at the end of the exploration after 651 collected data points (Fig. 6d), one can see that for this system, the reachable joint space set is not a rectangle as one might would expect.

The performance at the end of the exploration was evaluated on 100 joint configurations. In the shoulder joint the mean error was $0.40 \pm 0.09°$, in the elbow $0.31 \pm 0.01°$ (mean and standard deviation of three repeated experiments).

Note that while the true forward model for the simulated human arm is state independent, due to hysteresis and friction effects, the reported accuracy for the real system is only valid when starting from the rest position as in training. To investigate the decrease in performance, we selected 6 target configurations that are reached from the rest position and from the last configuration each. Here the error increases from $0.47 \pm 0.19°$ to $0.63 \pm 0.33°$ in the shoulder and from $0.25 \pm 0.14°$ to $0.46 \pm 0.43°$ in the elbow joint.

VII. Conclusion

In the present work, we formalized inverse model learning by optimizing an upper bound on the real performance error. Including this upper bound in the inverse learning objective turned out to be crucial. The proposed method to estimate the reachable set is not only an essential part in obtaining an inverse. Estimating the reachable set also enables to formulate the inverse model learning in an integral formulation, which showed to further increase the accuracy.

With the proposed active exploration strategy, which is based on maximizing a lower bound on real fill-distance, the necessary data to learn a forward model that is useful to learn the inverse could efficiently be generated in the high dimensional control input space.

Despite the fact that it is unrealistic to assume that the true forward model comes from the same RKHS as induced by the chosen kernel and its hyperparameters, in practice, the assumptions of our theoretical derivation are reasonably well fulfilled, implying that the derived bounds are actually useful in practice, can be computed easily and are tight.

Although the framework has been evaluated for control of musculoskeletal systems, it is a general method that can be applied to other inverse model learning settings as well. Since the learned inverse model provides an estimate of its performance, one could also embed our method into approaches that learn multiple paired forward-inverse models.

References

[1] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[2] B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters, "Learning inverse kinematics with structured prediction," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[3] M. Rolf, J. J. Steil, and M. Gienger, "Online Goal Babbling for rapid bootstrapping of inverse models in high dimensions," in *Proc. of the Int. Conf. on Development, Learning and Epigenetic Robotics*, 2011.

[4] B. Damas and J. Santos-Victor, "An online algorithm for simultaneously learning forward and inverse kinematics," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[5] M. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, 1992.

[6] D. Koert, G. Maeda, G. Neumann, and J. Peters, "Learning coupled forward-inverse models with combined prediction errors," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[7] J. Peters and S. Schaal, "Learning operational space control." in *Robotics: Science and Systems*, 2006.

[8] D. Driess, H. Zimmermann, S. Wolfen, D. Suissa, D. Haeufle, D. Hennes, M. Toussaint, and S. Schmitt, "Learning to control redundant musculoskeletal systems with neural networks and SQP: Exploiting muscle properties," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[9] M. Haruno, D. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural Computation*, 2001.

[10] R. Rayyes, D. Kubus, and J. Steil, "Learning inverse statics models efficiently with symmetry-based exploration," *Frontiers in Neurorobotics*, 2018.

[11] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *CoRR*, 2013.

[12] M. Rolf, "Goal babbling with unknown ranges: A direction-sampling approach," in *Proc. of the Int. Conf. on Development and Learning and Epigenetic Robotics (ICDL)*, 2013.

[13] D. Drieß, P. Englert, and M. Toussaint, "Constrained bayesian optimization of combined interaction force/task space controllers for manipulations," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[14] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[15] T. Hesselroth, K. Sarkar, P. P. Van Der Smagt, and K. Schulten, "Neural network control of a pneumatic robot arm," *IEEE Transactions on Systems, Man, and Cybernetics*, 1994.

[16] P. Van der Smagt and K. Schulten, "Control of pneumatic robot arm dynamics by a neural network," in *Proc. of the World Congress on Neural Networks*, 1993.

[17] Y. Cui, T. Matsubara, and K. Sugimoto, "Pneumatic artificial muscle-driven robot control using local update reinforcement learning," *Advanced Robotics*, 2017.

[18] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters, "Control of musculoskeletal systems using learned dynamics models," *Robotics and Automation Letters*, 2018.

[19] C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, and M. Asada, "Real-time inverse dynamics learning for musculoskeletal robots based on echo state gaussian process regression," in *Proc. of Robotics: Science and Systems*, 2012.

[20] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, 2004.

[21] D. Haeufle, M. Günther, A. Bayer, and S. Schmitt, "Hill-type muscle model with serial damping and eccentric force–velocity relation," *Journal of biomechanics*, vol. 47, no. 6, pp. 1531–1536, 2014.