A New Data Source for Inverse Dynamics Learning

Daniel Kappler*,1,2, Franziska Meier*,1,2,4, Nathan Ratliff[2] and Stefan Schaal[1,3]

Abstract—**Modern robotics is gravitating toward increasingly collaborative human robot interaction. Tools such as acceleration policies can naturally support the realization of reactive, adaptive, and compliant robots. These tools require us to model the system dynamics accurately – a difficult task. The fundamental problem remains that simulation and reality diverge–we do not know how to accurately change a robot's state. Thus, recent research on improving inverse dynamics models has been focused on making use of machine learning techniques. Traditional learning techniques train on the actual realized accelerations, instead of the policy's desired accelerations, which is an indirect data source. Here we show how an additional training signal – measured at the desired accelerations – can be derived from a feedback control signal. This effectively creates a second data source for learning inverse dynamics models. Furthermore, we show how both the traditional and this new data source, can be used to train task-specific models of the inverse dynamics, when used independently or combined. We analyze the use of both data sources in simulation and demonstrate its effectiveness on a real-world robotic platform. We show that our system incrementally improves the learned inverse dynamics model, and when using both data sources combined converges more consistently and faster.**

I. INTRODUCTION

Achieving reactive and compliant behavior is a cornerstone of robotic applications involving safe interaction with humans. A promising avenue for realizing reactive behaviors is the representation of motions through acceleration policies. Unfortunately, tracking accelerations is hard when we do not have an accurate model of the inverse dynamics of the system. Without a precise model of the systems dynamics, we typically resort to tracking desired trajectories–potentially generated by integrating the acceleration policy–employing feedback control to reject modeling errors of the dynamics. After careful tuning of the controller gains, this usually results in good tracking performance. However, we have to trade off compliancy and reactiveness of our controller against accuracy.

Thus, considerable effort has been put into developing machine learning methods that can learn or improve inverse dynamics models [1], [2], [3], [4]. These approaches attempt to identify a global inverse dynamics model, but collecting data that covers the full state-space is typically not considered a viable approach for high-dimensional systems. Furthermore, when considering motions with object interaction, such as pick and place tasks, learning one global model

*both authors contributed equally to this work
[1]AMD, MPI for Intelligent Systems, Tübingen, Germany
[2]Lula Robotics Inc, Seattle, USA
[3]CLMC Lab, University of Southern California, Los Angeles, USA
[4]RSE Lab, University of Washington, Seattle, USA

becomes even more involved, if not impossible, since the model has to be a function of contact and payload signals. Thus online learning has been a focus in these settings. However, online learning in this setting remains a challenge. The key difficulties are computationally efficient learning of models that are flexible enough to capture the non-stationary data of inverse dynamics mapping, and doing so on streaming data that is highly correlated.

To circumvent the issues of the aforementioned methods we follow the path of learning *task-specific* (error) models [5], [6], [7], [8]. This allows us to iterate collecting data specific to a task, learn an error model in offline fashion and then apply the learned model during the next task execution. While this approach is iterative by nature, we aim at making it as data-efficient as possible, such that only few iterations are required, while achieving consistent convergence in the error model learning process. Task specific models do not mitigate the problem of using payload signals or contextual information when trying to build a global model. However, it simplifies the overall global problem into two subproblems, finding a task specific inverse dynamics model and detecting which task model to use.

In this context a typical issue of traditional inverse dynamics learning approaches is that they actually learn the (error) model slightly off the desired trajectory. This comes from the fact that the data points used to learn the model are based on the actual achieved accelerations instead of the desired commanded accelerations. Thus if tracking is bad the collected data points are slightly off the desired trajectory for which we wanted to identify the inverse dynamics model. For instance, one extreme case of bad tracking is encountered when the system cannot overcome static friction. In that case, no useful data for inverse dynamics model learning is generated. Traditionally, this is circumvented by increasing the gains of the feedback control term, at the cost of compliancy. Alternatively, this issue is addressed by our recent work [9], which employs a *direct* loss function, minimizing the error between desired and actual accelerations, to learn feedback terms online.

In this work, we explore the intuition that feedback control can be viewed as an online technique to compensate for errors in a given a priori inverse dynamics model, as discussed in [9]. The feedback terms, compensate for errors between what a given model predicts and what we actually need. Therefore they naturally act as a convenient source of training data. We show how the *direct loss* on accelerations, as presented in [9], can be transformed into a loss on inverse dynamics torques, measured at desired accelerations. As a result, we now have two training data sources: the

traditional inverse dynamics training data points measured at actual accelerations, and this new training signal measured at commanded accelerations. We show how this additional data source leads to more consistent convergence of the task-specific inverse dynamics learning process.

In the following, we first review the inverse dynamics learning problem in Section II and discuss the differences between indirect and direct learning in this setting. In Section III we show how to combine the direct loss with the traditional indirect loss for inverse dynamics learning, followed by an overview of our complete task-specific learning approach in Section IV. Finally, we evaluate our proposed system in Section V.

II. BACKGROUND

Tracking desired accelerations with low feedback controller gains requires an accurate inverse dynamics model. The dynamics of a classical dynamical system can be expressed as

$$\tau = M(q)\ddot{q} + h(q, \dot{q}) \tag{1}$$

where $q, \dot{q}, \ddot{q}$ denote the joint positions, velocities and accelerations, M is the inertia matrix and h collects all the modeled forces such as gravitational, Coriolis, centrifugal forces, viscous and Coulomb friction. When possible, inverse dynamics approaches [10] model the system dynamics via the rigid body dynamics (RBD) equation of motions. Then, given sufficiently rich data, the RBD parameters can be identified using linear regression techniques [11], resulting in the approximate RBD dynamics model

$$\hat{\tau}_{\text{rbd}} = \hat{M}(q)\ddot{q}_d + \hat{h}(q, \dot{q}). \tag{2}$$

with approximate $\hat{M}$ and $\hat{h}$. This has been extended in [12], to additionally estimate payloads.

Unfortunately, the RBD model typically is not flexible enough to capture all non-linearities of the actual systems dynamics. As a result, the estimated RBD model is generally only a rough approximation. Thus, when attempting to track desired accelerations $\ddot{q}_d$ with $\hat{\tau}_{\text{rbd}}$ we achieve actual accelerations $\ddot{q}_a$ differing from $\ddot{q}_d$. Specifically when $\hat{\tau}_{\text{rbd}}$ is applied on the real system, with the true unknown dynamics model M, h, we can express the actual accelerations $\ddot{q}_a$ as

$$\ddot{q}_a = M(q)^{-1}[\hat{\tau}_{\text{rbd}} - h(q, \dot{q})] \tag{3}$$
$$= M(q)^{-1}[(\hat{M}(q)\ddot{q}_d + \hat{h}(q, \dot{q})) - h(q, \dot{q})]. \tag{4}$$

Note, if our estimated model $\hat{M}, \hat{h}$ were accurate, this expression would evaluate to $\ddot{q}_a = \ddot{q}_d$. However, this is typically not the case on real systems and because of this a feedback term τ_{fb} is required. This feedback term measures the error made and adds a corrective term to ensure accurate tracking. Traditionally, this feedback term is realized through PID control. The higher the gains, the better the tacking, at the cost of compliancy.

A. Learning Inverse Dynamics Models

To this end, various approaches to learning either the full inverse dynamics or an error model have been proposed. When learning an error model the total torque command is then a combination of any existing approximate model ($\hat{\tau}_{\text{rbd}}$), an error (torque) model (f_{iderr}) and a feedback term (τ_{fb}),

$$\tau_{\text{total}} = \hat{\tau}_{\text{rbd}} + f_{\text{iderr}} + \tau_{\text{fb}}. \tag{5}$$

One of the key challenges of inverse dynamics learning is computational efficiency. Predicting with learned models needs to be feasible within the real-time constraints of the systems consuming torque commands. Furthermore, it is typically assumed that the inverse dynamics mapping is non-stationary and can change over time. Thus, approaches that can incrementally learn and are computationally efficient enough for real-time deployment [1], [2], [3], [4], [5], [13] form one of the main research directions within the topic of inverse dynamics learning. However, learning globally valid models robustly on highly correlated data streams remains a challenge. More in depth discussion about the challenges and existing approaches can be found in [14], [15].

Some robustness can be achieved by using analytical (parametric) models such as RBD models as priors, and learning an error model on top of that [16], [17], [18]. Such approaches can revert to this prior knowledge, when the algorithm determines that the error model fit is uncertain.

Because of the difficulties of learning a globally valid model, some research has moved towards learning task/context specific inverse dynamics models [8], [6], [7], [19], [5], [20], [21]. In this setting it is feasible to collect task relevant data for offline learning, therefore simplifying the learning process. However, this comes at the cost of having to detect the correct context at run time such that the correct task model can be chosen. Our work, fits into this category, with the focus on effectively learning one task model.

Finally, learning inverse dynamics models is typically motivated by being able to use low-gain feedback control. However, traditional inverse dynamics learning approaches initially need high enough gains to achieve good tracking, such that relevant data is being generated. Little work has been done towards automatically lowering the feedback gains once a good model has been learned. An exception is work presented in [22] which allows for variable gains, using high gains, when the model is uncertain about its predictions and low gains when it is certain.

All of these methods learn inverse dynamics models on only the indirect data source measured at actual accelerations. In this work, we make use of two different data sources stemming from indirect and direct learning approaches to train inverse dynamics models.

Thus, before going into the details of our proposed approach, we discuss the notion of *indirect* and *direct* learning and present related work within that context.

B. Indirect vs Direct Learning of Inverse Dynamics

As mentioned above, most of the recent work on inverse dynamics learning can be classified as *indirect* learning

methods. The objective function that these methods optimize is given as

$$\mathcal{L}_{\text{indirect}}(\vec{w}) = \sum_{(x_a, \tau_{\text{total}}) \in \mathcal{D}} \|\tau_{\text{total}} - f(x_a; \vec{w})\| \quad (6)$$

where the input data point $x_a = (q, \dot{q}, \ddot{q}_a)$ is a combination of the state $(q, \dot{q})$ and the actual accelerations $\ddot{q}_a$. The output value τ_{total} is the corresponding applied torque that achieved the accelerations $\ddot{q}_a$. Here, the function that encodes the mapping from x to torques is defined as $f(x; \vec{w})$, where $\vec{w}$ are the open parameters of the chosen model.

Online, the robot attempts $\ddot{q}_d$ from state $(q, \dot{q})$. It calculates torque τ_{total} and observes true accelerations $\ddot{q}_a$. Rather than training at input point $x_d = (q, \dot{q}, \ddot{q}_d)$, the data point $x_a = (q, q, \ddot{q}_a)$ is used with target τ_{total}. The drawback to indirect learning is that convergence might be slow since the data is off the trajectory/distribution we want to optimize for. This is especially difficult when $\ddot{q}_a = 0$ due to static friction since many torques map to this value (i.e. the inverse function is not one-to-one at this point).

Alternatively, it was shown in [9] that it is possible to directly measure the gradient of the acceleration error of a system, which enables a new class of direct online learning algorithms. This approach aims to directly minimize the error between desired and actual accelerations, by optimizing the following *direct* loss function, for every pair of desired and actual accelerations,

$$\mathcal{L}_{\text{direct}}(\vec{w}) = \|\ddot{q}_d - \ddot{q}_a(\vec{w})\|_M^2 \quad (7)$$
$$= \|\ddot{q}_d - M^{-1}[\hat{\tau}_{\text{rbd}} + f_{\text{iderr}} - h]\|_M^2$$

where the actual accelerations $\ddot{q}_a$ are a function of the error model f_{iderr}. Note, traditional direct adaptive control methods [23], [24] have similar motivations – they use Lyapunov techniques to derive controllers that adjust the dynamics offset model with respect to some reference signal. In [9] we show how to effectively perform online learning on this objective leveraging well established online gradient descent techniques. However, this simple feedback term does not capture any structure of the error model and requires a relatively high learning rate to account for payload changes. In [13] we thus use [9] as a feedback term on acceleration errors and use the indirect loss function to learn a drifting Gaussian process to model larger structured inverse dynamics modeling errors. Note, [13] combines the direct and indirect learning as two separate learning processes with two different purposes: direct learning of an online adaptive feedback term and indirect learning of a locally valid inverse dynamics error model to capture larger errors. Also, [13] is a task independent online learning approach and theoretically applicable anywhere throughout the state space. However, on the flip side it retains no memory of previously learned error models, and thus is not able to improve over time.

Our work presented here is orthogonal to our previous work: We show that we can use direct and indirect learning within the same learning process of task-specific (feedforward) inverse dynamics (error) models, which can improve over time.

III. Combining Indirect and Direct Learning

To be able to learn the state-space dependent structure of the inverse dynamics modeling errors, we assume that the error model is identifiable in the space of $x = (q, \dot{q}, \ddot{q})$. Furthermore, since every task execution may slightly vary, we also follow an incremental learning process, meaning that each task execution generates a new training data set that can be used to update and improve our error model. Thus, our error models are indexed by k, indicating the kth learning iteration. For $k = 0$, meaning that no error model exists for the task at hand, we simply assume $f_{\text{iderr}}^0(x; \vec{w}^0) = 0$. Given this, the total torque applied to the system is the approximate rigid body dynamics model $\hat{\tau}_{\text{rbd}}(x_d)$ (if available) plus an offline learned error model f_{iderr}^k and a feedback term τ_{fb}:

$$\tau_{\text{total}} = \hat{\tau}_{\text{rbd}}(x_d) + f_{\text{iderr}}^k(x_d; \vec{w}^k) + \tau_{\text{fb}}. \quad (8)$$

Here we show how the direct and indirect loss functions can be combined into one loss function that uses two different data sources. In order to do so we 1) discuss the loss functions in the context of offline error model learning, 2) show that the two loss functions create two different training signals for the error model, and 3) use this result to combine direct and indirect learning.

A. Indirect Loss Function

We start out with discussing the details of learning an error model with an indirect loss. We compute the torque command based on the current state $(q, \dot{q})$ and desired accelerations $\ddot{q}_d$, apply this torque, and then measure actual accelerations $\ddot{q}_a$. Now we know what torque command achieves these measured accelerations and can use this data point to learn an inverse dynamics model. We collect all of these data points over the course of one task execution, for $t = 1 \ldots T$, such that we have T data points to learn parameters $\vec{w}^k$, initialized with the parameters $\vec{w}^{k-1}$.

In the indirect formulation, we try to optimize the parameters $\vec{w}^k$ such that the difference between the applied torque τ_{total} and the inverse dynamics model f_{id} at x_a is minimized:

$$\mathcal{L}_{\text{indirect}}(\vec{w}^k) = \sum_{t=1}^{T} \|\tau_{\text{total}}^t - f_{\text{id}}(x_a^t; \vec{w})\|^2 \quad (9)$$

Here we would like to utilize an approximate rigid body dynamics model (if available) and learn an error model f_{iderr} in order to optimize the f_{id} model. Notice, our approach does not require a rigid body dynamics model, all derivations hold when assuming a constant model $\hat{\tau}_{\text{rbd}} := 0$ as well. In this case we would learn the full inverse dynamics model, not using any domain specific knowledge. To compute what the RBD error is at input x_a^t, we have to evaluate $\hat{\tau}_{\text{rbd}}$ at x_a^t and subtract it from the total torque applied τ_{total}, such that, in the kth learning iteration, we optimize

$$\mathcal{L}_{\text{indirect}}(\vec{w}^k) = \sum_{t=1}^{T} \|\tau_{\text{total}}^t - \hat{\tau}_{\text{rbd}}(x_a^t) - f_{\text{iderr}}^k(x_a^t; \vec{w}^k)\|^2$$

Thus, using the indirect learning approach we optimize $f_{\text{iderr}}^k(x; \vec{w}^k)$ on the following data set

$$\mathcal{D}_{\text{indirect}}^k = \{x^t \leftarrow x_a^t, y^t \leftarrow \tau_{\text{total}}^t - \hat{\tau}_{\text{rbd}}(x_a^t)\}_{t=1}^{T}. \quad (10)$$

The quality of this training data set depends on how well we have tracked the task policy or trajectory. With accurate tracking behavior, one learning run should already give us a good approximation of the modeling errors. However, if tracking is bad, it very well may be that we require several learning iterations to estimate a good error model.

B. Direct Loss Function

To overcome the limitations of the indirect learning process, [9] proposes to use a direct loss (Eq. 7) to learn modeling errors. Here we use this loss in acceleration space [9], to derive an additional data source for inverse dynamics learning.

We start out with Eq. 7, drop the weighting of the acceleration error by the inertia matrix M, and instead multiply the accelerations with M

$$
\begin{aligned}
\mathcal{L}_{\text{direct}}(\vec{w}) &= \sum_{t=1}^{T} \| M\ddot{q}_d^t - M\ddot{q}_a^t(\vec{w}^k) \|^2 \qquad (11) \\
&= \sum_{t=1}^{T} \| (M\ddot{q}_d^t + h) - M\ddot{q}_a^t(\vec{w}^k) - h \|^2
\end{aligned}
$$

where we have also added and subtracted h. The true dynamics model M, h is never evaluated in our loss formulation, it is merely used to derive the direct loss formulation as shown in the following. We can now summarize the first term as the true rigid body dynamics model $\tau_{\text{rbd}}(\ddot{q}_d^t)$, evaluated at the desired accelerations, and we expand $\ddot{q}_a^t(\vec{w}^k)$ as follows

$$
\begin{aligned}
\mathcal{L}_{\text{direct}}(\vec{w}) &= \sum_{t=1}^{T} \| \tau_{\text{rbd}}(\ddot{q}_d^t) - M\ddot{q}_a^t(\vec{w}^k) - h \|^2 \qquad (12) \\
&= \sum_{t=1}^{T} \| \tau_{\text{rbd}}(\ddot{q}_d^t) - MM^{-1}[f_{id}(\ddot{q}_d^t; \vec{w}^k) - h] - h \|^2 \\
&= \sum_{t=1}^{T} \| \tau_{\text{rbd}}(\ddot{q}_d^t) - (\hat{\tau}_{\text{rbd}}(\ddot{q}_d^t) + f_{\text{iderr}}^k(x_d^t; \vec{w}^k)) \|^2 \\
&= \sum_{t=1}^{T} \| (\tau_{\text{rbd}}(\ddot{q}_d^t) - \hat{\tau}_{\text{rbd}}(\ddot{q}_d^t)) - f_{\text{iderr}}^k(x_d^t; \vec{w}^k) \|^2
\end{aligned}
$$

where $f_{id}(\ddot{q}_d^t)$ represents the state based rigid body dynamics and error model without a feedback term which should ideally be zero. We now have transformed the loss on accelerations to a loss on torque commands at the input point x_d^t. Note that this transformed loss intuitively means that we want to minimize the difference between our error model $f_{\text{iderr}}^k(x_d^t; \vec{w}^k)$ and the true modeling error $\tau_{\text{error}}^t = (\tau_{\text{rbd}}(\ddot{q}_d^t) - \hat{\tau}_{\text{rbd}}(\ddot{q}_d^t))$ at input $x_d^t = (q^t, \dot{q}^t, \ddot{q}_d^t)$. While intuitively pleasing, we unfortunately do not have access to the true modeling error τ_{error}^t. However, we can get an estimate of the modeling error $\hat{\tau}_{\text{error}}^t = \tau_{\text{fb}}^t + f_{\text{iderr}}^{k-1}(x_d^t; \vec{w}^{k-1})$ by combining the feedback term τ_{fb}^t and the error model f_{iderr}^{k-1} from the previous task execution (similar to feedback error learning [24]), which results in the following loss

$$
\mathcal{L}_{\text{direct}}(\vec{w}) = \sum_{t=1}^{T} \| \hat{\tau}_{\text{error}}^t - f_{\text{iderr}}^k(x_d^t; \vec{w}^k) \|^2 \qquad (13)
$$

Similar to the indirect learning we can now construct a data set

$$
\mathcal{D}_{\text{direct}}^k = \{ x^t \leftarrow x_d^t, y^t \leftarrow \tau_{\text{fb}}^t + f_{\text{iderr}}^{k-1}(x_d^t; \vec{w}^{k-1}) \}_{t=1}^{T} \qquad (14)
$$

which can be used to learn or update the new error model f_{iderr}^k. We now receive data points directly on the desired accelerations. However, also this data set's quality depends on tracking accuracy. With low feedback gains, the initial

τ_{fb} may not really capture the errors very well, such that the first learning iteration may only capture part of the modeling errors.

Thus, with low feedback gains, we may require multiple learning iterations to learn an accurate error model. Whereas increasing the feedback gains g would lead to improved tracking, increasing the fidelity of the data, at the cost of compliancy. Here we simply propose to do both: use g_{low} to compute the feedback terms $\tau_{\text{fb}}(g_{\text{low}})$ which are sent to the system, and use g_{high} to compute feedback terms $\tau_{\text{fb}}(g_{\text{high}})$ which are sent to the learner. Notice, the feedback term using $\tau_{\text{fb}}(g_{\text{high}})$ is never applied on the system, thus, we maintain a very compliant system while obtaining better error data for the portion of the state space reached with the g_{low}. This can be helpful to break stiction or counteract high friction with f_{iderr} after fewer iterations which otherwise would not be possible with traditional inverse dynamics learning approaches and low gains.

C. Joint Inverse Dynamics Learning

The key insight of this paper is that we can use the feedback term as an error estimate for the desired accelerations. Thereby the error model learning problem has two data sources *indirect* and *direct*. Both exhibit the same structure to optimize the error model. Hence, we can formulate a joint function approximation problem of the form:

$$
\mathcal{L}_{\text{joint}}(\vec{w}^k) = \sum_{(x,y) \in \mathcal{D}_{\text{joint}}^k} \| y - f_{\text{iderr}}^k(x; \vec{w}^k) \| \qquad (15)
$$

where data points for the actual accelerations $\ddot{q}_a^t$ for every timestep t can be used as well as data points for the desired accelerations $\ddot{q}_d^t$ as described by

$$
\mathcal{D}_{\text{joint}}^k = \mathcal{D}_{\text{direct}}^k \cup \mathcal{D}_{\text{indirect}}^k. \qquad (16)
$$

IV. TASK SPECIFIC INVERSE DYNAMICS LEARNING

Implementing our approach required several design decisions on the levels of motion generation, control and learning. Here we will give a short overview of our design choices, and give an algorithmic overview of our iterative approach to learning task-specific inverse dynamics models.

On the motion generation level, we assume that a kinematic policy for our task is provided, meaning that we can obtain desired accelerations $\ddot{q}_d^t$ for every state $q^t, \dot{q}^t$ relevant to our task. In particular, we use kinematic Linear Quadratic Regulators (LQRs) [25] to provide us with the acceleration policy that is being executed. On the control level we use two types of feedback controllers: traditional PID control and the recently introduced adaptive feedback learning (DOOMED [9]). When tuned sufficiently well, both approaches can result in good tracking performance. We also have to choose a function approximator for the error model. Because we are learning the error model offline, learning speed is not our main concern. However, prediction speed is important since the models need to be evaluated at 1000Hz (hard real-time) on our robotic platform. We choose a simple feedforward neural network which is capable of learning

Algorithm 1 Execute Task and Collect Data

Require: f_{iderr}^{k-1}, $\pi(q^t, \dot{q}^t)$, system(τ^t), $q^0, \dot{q}^0$
1: $\mathcal{D}_{\text{joint}}^k = \varnothing; t = 0$
2: **while** not converged **do**
3:     $\ddot{q}_d^t = \pi(q^t, \dot{q}^t)$
4:     $x^t = (q^t, \dot{q}^t, \ddot{q}_d^t)$
5:     $\tau^t = \hat{\tau}_{\text{rbd}}(x_d) + \tau_{\text{fb}}^t + f_{\text{iderr}}^{k-1}(x_d^t; \vec{w}^{k-1})$
6:     $t = t + 1$
7:     $q^t = \text{system}(\tau^{t-1})$
8:     $\dot{q}^t, \ddot{q}_a^{t-1} = \text{finiteDiff}(q^t, q^{t-1})$
9:     update feedback term τ_{fb}^t
10:    $\mathcal{D}_{\text{joint}}^k = \mathcal{D}_{\text{joint}}^k \cup (x_a^{t-1}, \tau^{t-1} - \hat{\tau}_{\text{rbd}}(x_a^{t-1}))$
11:    $\mathcal{D}_{\text{joint}}^k = \mathcal{D}_{\text{joint}}^k \cup (x_d^{t-1}, \tau_{\text{fb}}^{t-1} + f_{\text{iderr}}^{k-1}(x_d^{t-1}; \vec{w}^{k-1}))$
12:
13: **end while**
14: **return** optimize$(f_{\text{iderr}}^k, \mathcal{D}_{\text{joint}}^k)$

---

nonlinear mappings and predictions require only a simple feedforward pass.

Finally our iterative learning approach (Algorithm 1) can be summarized as follows:

1) task execution: Run task with approximate RBD model, feedback control, and feedback error model if existent (line 5). Construct data set's with two different sources of information (equation (10) (line 10) and (14)) (line 11) during the task execution, as shown in Algorithm 1.
2) learning phase: Update the error model function approximators based on new data points, or construct and initialize the error models if none exist (line 14).
3) repeat the task.

As we empirically show in the Section V, exploiting both sources of information allows us to obtain a better fit of the error model with less task iterations, compared to using a single data source.

V. EXPERIMENTS

We evaluate our approach in two different settings. First, we analyze the proposed usage of indirect and direct data sources in order to learn a task specific inverse dynamics error model on a 2D simulation. This allows us to extensively test characteristics of the learning problems based on the different data sources, using the same function approximator, under various simulated noise levels, stictions, frictions, and a wrong RBD model. This evaluation is focused on investigating the importance and influence of the different data sources rather then the function approximator itself. Second, we report results on task specific inverse dynamics learning using both data sources on the KUKA lightweight arm of our robotic platform shown in Figure 3. We start out describing the evaluation of our 2D simulation setting.

A. Simulation Experiments

1) System description: This evaluation is based on a simple 2D example [26] of a system for which the approximate RBD dynamics model differs drastically from the true dynamics. In this simulated system the true mass is

set to $\mathbf{M} = 5I$, while the approximate mass is assumed to be $\mathbf{M} = 0.5I$. The system attempts to realize a simple acceleration policy, defined as a PD-controller with desired state at $q_{\text{des}} = (1, 1)$ and initial state $q_{\text{init}} = (0, 0)$ and $\dot{q}_{\text{init}} = (0, 0)$. Furthermore, we simulate the true system to experience friction and stiction which is not modeled by the approximate RBD model. Finally we also add sensing noise to the position trajectories to mimic noisy sensor measurements. The source code of our example simulation with all parameters, friction and stiction models used for the experiments can be found at [26].

2) Details of error model learning: Every experiment involves running the iterative learning process for 20 iterations. After each cycle a neural network is trained on the collected data and is then used to predict the modeling error in the next cycle. The error model f_{iderr} is learned via a neural network structure which consists of fully connected layers (200, 100, 50, 20, 1) with non-linearities (prelu [27]) after every layer except the last. We optimize one network per simulated joint of our system. The neural network is trained on the indirect data set (equation (10)), the direct data set (equation (10)), and as proposed in this paper the joint data set (equation (16)).

3) Experimental Setup: To evaluate the use of the data source variants (direct, indirect or joint) in a principled manner, we simulate various system conditions:

- 4 maximum sensing noise levels in meters are reported: low (0.0001), medium (0.0005), high (0.007), very high (0.008)
- 2 friction levels: medium and high
- 2 stiction levels: medium and high

across different hyper-parameter settings:

- 2 different number of training epochs of the neural network training per task iteration are evaluated (20, 50).
- 2 different gain settings (applied) for DOOMED and PID: low and high. The PID gains have been tuned such that even without error model we achieve convergence to the goal. Low gains are one order of magnitude lower.

The friction model is discontinuous and changes throughout the state space. The feedback term for learning is exponentially filtered with the same value (0.1) for all experiments. This is possible since the learning feedback term is not applied to the system.

For the detailed noise, friction and stiction models as well as the exact parameters to replicate the experiments please check out [26]. For each data source variant (direct, indirect, joint) a total of 16 system combinations were executed, 10 times each, with different random seeds. A total of 1280 experiments were performed to cover the different hyper-parameter settings. To make results comparable, the random seed was kept consistent across the data source variants. For each of these runs we record the average magnitude of the applied feedback torque, the desired and actual acceleration, as well as the desired and actual position.

| (a) PID, low system noise | (b) PID, very high system noise | (c) doomed, low system noise | (d) doomed, very high system noise |

Fig. 1: Hand-picked simulation runs to illustrate (dis)-advantages of the *direct* or *indirect* data sources. On the top row we show position tracking error as a function of learning iterations, for both low-gain PID and low-gain DOOMED feedback control on systems with low and very high noise. In low noise settings, for both PID and DOOMED, the direct data source leads to improved position tracking with increasing number of learning iterations. However, when using the indirect data source the NN cannot capture the error model, and thus position tracking does not improve at all (with PID) or more slowly (with doomed) over time. In the very high noise setting, the learning convergence with indirect data source is comparable to the low noise setting, but when using the direct data source alone, we see erratic tracking convergence. The direct data is affected more by the noise. For both PID and DOOMED, the position tracking performance converges more consistently when using the joint data set.

4) Illustration of Indirect vs Direct: We start out with illustrating some hand-picked scenarios that showcase the differences of learning on indirect vs direct data in Figure 1. We choose examples with the same parameter settings, and illustrate the difference when going from low to high noise. For the chosen examples, the learning rates/feedback gains were set to the low value, such that friction and stiction was not so easily overcome with feedback terms alone. This has the effect that for low-gain PID control we basically see no improvement in position tracking over time when using the indirect data only. With DOOMED, some improvement can be observed (with indirect data) - but at a slower rate compared to using direct data. The noise level does not affect the indirect learning process much. However, it affects the learning on direct data. In the low noise setting we observe how position tracking improves over time, but in the high noise setting this is not true. However, when combining both data sources, we get consistently good convergence of position tracking performance, even with low-gain feedback control.

5) Extensive Evaluation: To provide a more extensive evaluation we now present results obtained when averaging across all system and parameter settings, see Fig. 2. These results show that, on average, using both data sources results in more consistent and faster convergence of the position tracking error, when compared to using the traditional approach of using indirect data alone. Specifically, in the case of low-gain PID control, the error model trained with the indirect data alone does not improve the position tracking error, whereas the joint learning process does.

In the high-gain setting, the improvements are less pronounced. However, we want to stress that one important goal of this work is to learn an accurate task specific inverse dynamics model, while being as compliant as possible. Thus, the high gain setting shows that our proposed approach does not deteriorate in case of a less compliant system configuration, but there is not much to gain in using the additional data source. Intuitively, this makes sense, since in a high gain setting, the feedback control term is expected to provide good tracking performance in the very first task execution already. Thus the indirect data collected during that first run already provides very good data to learn a model for that particular task. We want to emphasize that there exist parameterizations and system settings that can lead to better performance by a single data source. However, on average, the direct data source seems to be most sensitive to the system/parameter combination, and the indirect data source requires higher feedback gains to be useful. The joint data source, however, can achieve superior and more consistent model learning performance, in low-gain settings.

B. Real robot experiments

We have evaluated our method with two real world robot experiments, a quantitative and a qualitative one. Both experiments were performed on the platform shown in Fig. 3 (top). Our platform consists of two KUKA lighweight arms, each of which has 7 degrees of freedom. All experiments were performed on one arm resulting in a 21-dimensional input for the error model learning problem. The control system operates on a hard real-time loop of 1 kHz, thus, all predictions are performed in less than 1 ms. All our experiments presented here, use DOOMED as feedback controller. We optimize one neural network per joint with the following structure: 4 fully connected layers (200, 100, 50, 1) with non-linearities (prelu [27]) after every layer except the last. Furthermore, we bound the predicted torques to $\pm 20 Nm$. For both real robot experiments we analyzed the task-specific inverse dynamics learning based on the joint data set, since this has shown to provide the most consistent performance.

For our quantitative experiments we used a pre-planned sequence of LQR policies to generate desired accelerations. We execute the task 10 times, always starting in the same position, up to the precision of the arm. After each run, the collected data was used to re-optimize the neural networks. Fig. 3 shows how the sum of squared feedback terms (τ_{fb}), averaged across all joints, changes with each run. Notice, the first run uses no error model, thus reflecting how much the

| (a) PID: low-gain | (b) PID: high-gain | (c) doomed: low-gain | (d) doomed: high-gain |

Fig. 2: Average results for low-gain and high-gain feedback control, and averaged across all system and parameter settings. (top row) shows the position tracking error convergence as a function of the number of learning iterations. (bottom row) shows the average feedback term applied. We plot the mean and the mean plus one standard deviation of the results. *low-gain:* In this setting, even when averaging across all system settings and parameters, we observe similar position tracking behavior for the data source variants as in our hand-picked illustrations. Indirect data alone is not able to capture the error model, which also explains why higher feedback torques are required in this setting. Using direct data alone, results have a higher variance. On average the mean tracking behavior degrades again after a few learning iterations (PID) or is somewhat erratic (DOOMED). Using the joint data set results in the most consistent tracking error convergence for both feedback controllers. *high-gain* When using high feedback gains, the joint data set method does not gain as much in convergence performance (over using indirect data alone). However using both data sources also does not degrade performance.

feedback term has to compensate for the modeling errors. Within one iteration we are already able to capture most of the error with the learned error model. Hence, our learned model in combination with the rigid body dynamics model is now a reliable inverse dynamics model for this task. We want to stress that after every trial the newly obtained data is used to further refine our task specific error model, and despite the data correlation between trials our updated model does not degrade. In Fig. 4 we show the feedback term trajectory for the first and last task execution, per joint. Again it can be seen how our learned model compensates for the errors such that the feedback term only has to adjust for system noise.

In our qualitative example we show the data efficiency of our approach for a real world manipulation task. The robot has to pick up a heavy drill from a table and place it on a different location on the same table. Since we are interested in collaborative setting, we chose the learning rate of DOOMED as low as possible such that the robot itself is as compliant as possible while still being able to at least lift the drill. As shown in the video at [28], the usage of the joint data set, enables our system to significantly improve the performance of the pick and place task after a single iteration.

VI. DISCUSSION

In this work, we have proposed to use two different data sources to learn inverse dynamics models. We have evaluated the usage of both data sources, indirect and direct, both in simulation and on a real system. Our evaluations demonstrate that combining the indirect and direct data leads to more consistent and often faster learning convergence,

Fig. 3: (top) Our robot platform is shown in the top figure. (bottom) Mean squared feedback terms (τ_{fb}) as a function of learning iterations. Run 1 corresponds to Fig. 4 (top) and run 10 to Fig. 4 (bottom).

compared to using the traditional indirect data source only. Furthermore, this superior performance of the combined data set is especially noticeable in the low-gain feedback control setting, such that we can effectively learn error models while being more compliant from the beginning.

Nevertheless some restrictions to our approach exist: The proposed method is based on the assumption that we do not visit the same part of the state space $(q, \dot{q})$, with differing amounts of payload. Since the input of the error model does not contain any information about the payload change of the

Fig. 4: (top) The initial feedback terms (τ_{fb}), per joint, without a learned error model. (bottom) τ_{fb} after 10 learning iterations.

system other then the generated feedback τ_{fb}, such a task would lead to data ambiguities. For example, picking up a drill from a position on a table, placing it somewhere else and repeating the pickup without the drill could not be expressed by a single task-specific inverse dynamics model right now. A potential extension to alleviate this problem would be to provide additional inputs, either sensing, or more abstract provided information to the learning system.

For this work we do not analyze how the system performs when it is strongly perturbed. This could result in undesirable predictions since the error model has not been trained on any data for that input space. This is however a general problem for task specific models. We want to emphasize that we have bounded the output of the error model to a reasonable torque limit for our system, but this limit has not been exceeded during our experiments. Furthermore, we believe that this problem can be addressed in future work as discussed in the following section.

VII. CONCLUSIONS AND FUTURE WORK

It is important to note, that in individual experiments, error models trained on the direct, indirect and joint data have all shown superior performance for certain system and learning parameterizations. Overall training on the joint data set results in more consistent and faster convergence and lower exerted torques. However, an interesting direction for future work is to exploit the structure in the different data sources in order to identify which source is more reliable. Ideally this will allow to train an error model for which the performance is always at least as good as the better one of the two data sources.

We want to further investigate the scheduling of the gains of the feedback term based on the performance of the error model. This should enable our system to increase the

level of compliance even more over time. Being able to lower the gains also enables better detection of structured perturbations, e.g. when a user is pushing the robot arm. The main reason for that is that our learned error model can capture the modeling errors of the rigid body dynamics and the feedback term only has to correct for sensor noise. Therefore, it is possible to detect otherwise difficult problems such as collision with objects.

To the best of our knowledge the usage of the two data sources is a novel approach to inverse dynamics learning. We have empirically shown, that even with a low-gain feedback controller, this can lead to consistent and fast error model learning. The effectiveness of this approach has further been evaluated on a real system.

VIII. ACKNOWLEDGMENTS

REFERENCES

[1] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: Incremental real time learning in high dimensional space," in *Proc of. the International Conference on Machine Learning (ICML)*, 2000, pp. 1079–1086.

[2] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, "Local Gaussian process regression for real time online model learning," in *Proc of. Neural Information Processing Systems (NIPS)*, 2008, pp. 1193–1200.

[3] A. Gijsberts and G. Metta, "Real-time model learning using incremental sparse spectrum Gaussian process regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[4] F. Meier, P. Hennig, and S. Schaal, "Incremental Local Gaussian Regression," in *Proc of. Neural Information Processing Systems (NIPS)*, 2014.

[5] L. Jamone, B. Damas, and J. Santos-Victor, "Incremental learning of context-dependent dynamic internal models for robot control," in *Proc. of the IEEE International Symposium on Intelligent Control (ISIC)*, 2014.

[6] M. Toussaint and S. Vijayakumar, "Learning discontinuities with products-of-sigmoids for switching between local models," in *Proc of. the International Conference on Machine Learning (ICML)*, 2005.

[7] G. Petkos, M. Toussaint, and S. Vijayakumar, "Learning multiple models of non-linear dynamics for control under varying contexts," in *International Conference on Artificial Neural Networks*. Springer, 2006, pp. 898–907.

[8] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural networks*, vol. 11, no. 7, 1998.

[9] N. Ratliff, F. Meier, D. Kappler, and S. Schaal, "Doomed: Direct online optimization of modeling errors in dynamics," *Big Data*, vol. 4, no. 4, pp. 253–268, 2016.

[10] J. J. Craig, P. Hsu, and S. S. Sastry, "Adaptive Control of Mechanical Manipulators," *The International Journal of Robotics Research (IJRR)*, 1987.

[11] C. H. An, C. G. Atkeson, and J. M. Hollerbach, "Estimation of inertial parameters of rigid body links of manipulators," in *Proc of. the IEEE Conference on Decision and Control (CDC)*, 1985, pp. 990–995.

[12] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Rigid body load identification for manipulators," in *Proc of. the IEEE Conference on Decision and Control (CDC)*, vol. 24, 1985, pp. 996–1002.

[13] F. Meier, D. Kappler, N. Ratliff, and S. Schaal, "Towards robust online inverse dynamics learning," in *Proc of. the IEEE/RSJ Internationl Conference on Intelligent Robots and Systems (IROS)*, Oct 2016.

[14] O. Sigaud, C. Salaün, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, 2011.

[15] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[16] D. Nguyen-tuong and J. Peters, "Using Model Knowledge for Learning Inverse Dynamics," *Proc of. the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[17] J. S. de la Cruz, E. Calisgan, D. Kulić, W. Owen, and E. A. Croft, "On-line dynamic model learning for manipulator control," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 869–874, 2012.

[18] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental semiparametric inverse dynamics learning," in *Proc of. the IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 544–550.

[19] T. Petrič, A. Gams, L. Žlajpah, and A. Ude, "Online learning of task-specific dynamics for periodic tasks," in *Proc of. the IEEE/RSJ Internationl Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 1790–1795.

[20] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters, "Learning inverse dynamics models with contacts," in *Proc of. the IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3186–3191.

[21] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[22] N. T. Alberto, M. Mistry, and F. Stulp, "Computed torque control with variable gains through gaussian process regression," in *Proc of. the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 212–217.

[23] K. J. Astrom and D. B. Wittenmark, *Adaptive Control*, 2nd ed. Dover, 2008.

[24] "Feedback error learning and nonlinear adaptive control," *Neural Networks*, vol. 17, no. 10, pp. 1453 – 1465, 2004.

[25] R. Stengel, *Optimal Control and Estimation*. Dover, New York, 1994.

[26] D. Kappler and F. Meier, "The code used to obtain the simulation results." https://github.com/dkappler/idsim.git, 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[28] D. Kappler and F. Meier, "Video showing the qualitative experimental results." https://vimeo.com/182948172, 2017.