Accelerating model learning with inter-robot knowledge transfer

Ndivhuwo Makondo[1], Benjamin Rosman[2] and Osamu Hasegawa[3]

***Abstract*— Online learning of a robot's inverse dynamics model for trajectory tracking necessitates an interaction between the robot and its environment to collect training data. This is challenging for physical robots in the real world, especially for humanoids and manipulators due to their large and high dimensional state and action spaces, as a large amount of data must be collected over time. This can put the robot in danger when learning *tabula rasa* and can also be a time-intensive process especially in a multi-robot setting, where each robot is learning its model from scratch. We propose accelerating learning of the inverse dynamics model for trajectory tracking tasks in this multi-robot setting using knowledge transfer, where robots share and re-use data collected by pre-existing robots, in order to speed up learning for new robots. We propose a scheme for collecting a sample of correspondences from the robots for training transfer models, and demonstrate, in simulations, the benefit of knowledge transfer in accelerating online learning of the inverse dynamics model between several robots, including between a low-cost Interbotix PhantomX Pincher arm, and a more expensive and relatively heavier Kuka youBot arm. We show that knowledge transfer can save up to 63% of training time of the youBot arm compared to learning from scratch, and about 58% for the lighter Pincher arm.**

I. INTRODUCTION

To control a robot manipulator to track a specified trajectory, model-based control offers many advantages over traditional PID-based control, including potentially higher tracking accuracy, lower energy consumption and lower feedback gains – which results in more compliant and reactive control [1]. The model is used to predict, for example, the joint torques given the desired trajectory in terms of joint positions, velocities and accelerations. However, the performance of model-based control relies heavily on the accuracy of the models used in capturing the dynamics of the real system under control and its environment. The dynamics model can be developed from first principles in mechanics, based on the Rigid Body Dynamics (RBD) framework [2], resulting in a parametric model, with parameters such as the inertial parameters of link mass, center of mass and moments of inertia, and friction parameters, that must be estimated precisely.

In practice, however, it is difficult to obtain a sufficiently accurate dynamics model for many modern robotic systems based on the parametric RBD model, due to unmodeled non-linearities such as friction, backlash and actuator dynamics. Thus, several assumptions are made to simplify the process, such as rigidity of links or that friction has a simple analytical form, leading to inaccuracies in the model. The inaccurate model can lead to large tracking errors, which must be compensated for using high-gain PID control. As high-gain control would turn the robot into a danger for its environment in reducing compliance, more accurate models are needed.

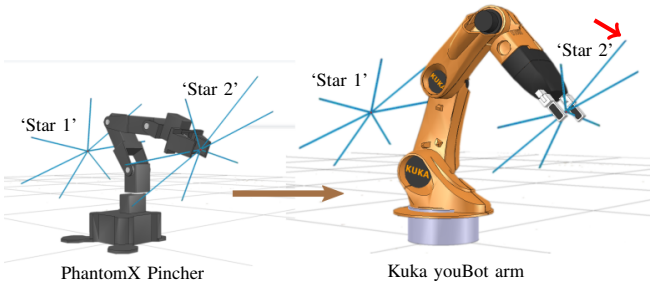


Fig. 1: Example of knowledge transfer from a low-cost robot to a more expensive and heavier robot. The red arrow indicates a point that is challenging for the robot to reach at high speeds.

Learning robot models for control based on regression techniques is typically employed as an alternative in such cases where the physical parameters of the robot are unknown or inaccurate. Unknown non-linearities can be taken into account as the model is estimated directly from measured data [3]. Furthermore, as learning-based techniques are capable of modeling complex systems, modern robots can be designed to suit various demands and environments, rather than to simplify modeling. As a result, learning approaches have attracted much interest recently.

Despite the widespread adoption of state-of-the-art machine learning techniques to robotics, learning for real-time physical systems is still a challenging problem, mainly due to the large and high-dimensional state and action spaces of humanoids and manipulators. Furthermore, their physical embodiment allows only a limited time for collecting training data. For example, real-time learning of the inverse dynamics model for manipulator trajectory tracking tasks necessitates an interaction between the robot and its environment to collect training samples along the desired trajectory over many trials. This can be time-consuming for manipulators and humanoids.

This is worse when we have multiple, kinematically and dynamically different robots, that are learning models

[1]Ndivhuwo Makondo is with the Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Japan; and the Mobile Intelligent Autonomous Systems, Council for Scientific and Industrial Research, South Africa nmakondo@csir.co.za

[2]Benjamin Rosman is with the Mobile Intelligent Autonomous Systems, Council for Scientific and Industrial Research, South Africa; and the School of Computer Science and Applied Mathematics, University of the Witwatersrand, South Africa brosman@csir.co.za

[3]Osamu Hasegawa is with the Faculty of Engineering, Dept. of Systems and Control Engineering, Tokyo Institute of Technology, Japan; and SOINN Inc. oh@soinn.com

through the same laborious process. Thus, in this paper, we aim to accelerate learning for a *new robot* by re-using data generated by a pre-existing robot for the *same task*, as an additional data set for the new robot, as illustrated in Fig. 1. By re-using data generated by other robots as a means of knowledge transfer we show that learning for the new robot will be biased towards relevant spaces such that fewer trials of interacting with the environment are needed, thus improving the learning speed.

We propose a scheme for generating training data online from the robots, which we use to learn models by which the knowledge is transferred between them, and, in contrast to previous work in transfer learning for trajectory tracking, we demonstrate the benefit of knowledge transfer for accelerating online learning of the inverse dynamics model. We analyze learning for several different robots and discuss various aspects of knowledge transfer between the robots.

II. Related Work

A. Model Learning for Control

In this paper, our focus is on model-based control for robot manipulators, specifically inverse dynamics for trajectory control tasks. The inverse dynamics model is usually learned using state-of-the-art non-parametric regression techniques. Real-time online learning of this model allows adaptation to changes in the robot dynamics, load, or actuators, and can be broadly broken down into two categories: global methods and local methods. Global methods model a regression function that is defined globally in input space; whereas local methods seek to partition the input space into smaller regions and define a function only valid locally for each region. Examples of global methods include those that make use of the entire available data set, such as those based on deep learning methods [4], random features and Ridge Regression [5], and methods that make use of a sparse set representing the input space [6], [7], [8].

Local methods are inspired by the idea of locally weighted learning (LWL) for control [9] and include techniques that build locally linear models such as Locally Weighted Projection Regression (LWPR) [10], locally non-linear models such as Local Gaussian Processes (LGP) [11] and Local Gaussian Regression (LGR) [12], and Local online Support Vector Regression (LoSVR) [13]. Drifting Gaussian Processes is also a local model, specifically aimed at streaming data [14].

Learning using any of the techniques presented above requires that the robot is first operated in order to collect samples, and this can be time-consuming for manipulators and humanoids. The next section reviews techniques that can be used to accelerate learning under various circumstances.

B. Accelerating Model Learning

The learning algorithms presented above disregard any prior knowledge about the robot system that may be available, such as the potentially inaccurate RBD model, or some parts of it (e.g., gravity component), and so begin learning from scratch. One recent technique for accelerating learning is marrying the physics-based RBD model (if available) with non-parametric learning models into a semi-parametric model. The benefits include faster learning speeds, higher accuracy, and better generalization [15]. The parametric RBD model component acts as prior knowledge and is defined over the entire state space, and the non-parametric component models the non-linearities and adapts to changes online.

Examples include techniques that incorporate the parametric RBD model into the non-parametric part as a mean function [15], kernel function [16], and those that use first order approximations of the RBD equation to initialize the local models of the LWPR model [17]. Other techniques instead model the inverse dynamics error (or residual), using random features and Recursive Regularized Least Squares (RRLS) [18], or as a constant offset that is continuously adapted via online gradient descent to minimize the error [19].

Another set of approaches for accelerating learning of inverse dynamics, which is of particular interest to our work, is based on the concept of transfer learning [20], where knowledge gained while solving a task in one domain is leveraged to help improve learning a new task in another domain. Transfer learning approaches for robotics can be broadly broken down into two categories: *inter-task transfer* and *inter-robot transfer*.

In inter-task transfer, a robot leverages knowledge of previous tasks to speed up learning a *new related task*. Our work falls under inter-robot transfer, where a data set generated by one robot (a *source robot*) performing a task is used to aid learning of the *same task* by a *new robot* (a *target robot*). In general the source and target robots may have different kinematic and dynamic properties, so the source data must be mapped into the domain of the target robot for it to be useful. Not much work dealing with this case exists for trajectory tracking problems and the majority of those available model this mapping using manifold alignment techniques [24].

A manifold alignment based approach has been used to show the possibility of transfer for inverse dynamics between two simulated robot manipulators [25]. In [26], they argued that an optimal map between dynamic systems is a dynamic map, and they applied their proposed dynamic map on planar arms and two different quadrotor platforms, where the systems were modeled as single-input, single-output (SISO) systems, which has not been shown to generalize to high-dimensional manipulators.

In some cases it has been shown that learning can be accelerated by initializing the model with random data generated through a motor babbling process [28], [29]. Here, the robot tracks random joint trajectories using a PID controller while the model is updated using the generated data.

In this work, we investigate the acceleration of learning the inverse dynamics model for trajectory tracking using inter-robot transfer. Based on our review this approach has not received much attention. In [25] they demonstrated the possibility of accelerating learning between two robots. However, their aim was to show the soundness of transfer, and so they assumed that analytical models of the robots

are available for data collection. This assumption is limiting, since the purpose of learning is to obtain such models from robot data.

In a realistic scenario, an approach for collecting training data from robots *without full knowledge* of the robot models is needed. The contributions of this paper are thus i) an approach which leverages motor babbling techniques to generate such training data from the robots without assuming knowledge of robot models, and ii) a demonstration of the benefit of transfer between several robots with different kinematic and dynamic properties for trajectory tracking tasks.

Other work in improving learning of dynamic models using knowledge transfer exists. In particular, approaches based upon ideas from adaptive control have received considerable attention recently, including work that transfers knowledge from a simulation to the physical system [21] and between systems with different sizes and dynamic properties [22], [23]. Although these approaches do not rely on correspondences between systems, they assume that the systems have similar state and/or action spaces. In contrast, our approach is applicable to the general case of systems with different state and action spaces.

III. Problem Statement

The inverse dynamics model of a robot manipulator relates the joint positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and accelerations $\ddot{\mathbf{q}}$ with the corresponding forces and torques $\boldsymbol{\tau}$ required to follow a specified joint-space trajectory; and can be described analytically using the well-known Rigid Body Dynamics formula

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}), \tag{1}$$

where $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{g}$ are the inertial, Coriolis, and gravitational terms, respectively. The feed-forward torque $\boldsymbol{\tau}_{ff}$ for the current desired state $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$ is predicted using Eq. 1, while a feedback torque $\boldsymbol{\tau}_{fb}$, computed using a feedback controller (e.g. a simple PID controller), is used to stabilize the system. Therefore the total torque applied to the robot is $\boldsymbol{\tau}_a = \boldsymbol{\tau}_{ff} + \boldsymbol{\tau}_{fb}$.

Unfortunately, this formulation is limited when used to control modern robotics systems, due to issues discussed in Section I. Thus, learning the model from data generated by the robot, using non-parametric machine learning techniques has emerged as an alternative. Here, the problem is reduced to a standard supervised learning setting

$$\boldsymbol{\tau} = \mathbf{D}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) + \epsilon, \tag{2}$$

where we seek to learn the dynamic model $\mathbf{D}(\cdot)$ from input-output pairs $\{(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}), \boldsymbol{\tau}\}$ generated by the robot, and $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is the output noise modeled as Gaussian noise with zero mean and variance σ^2.

Due to the large state and action spaces of most industrial manipulators, learning this model is done online along specified desired trajectories, resulting in a trajectory-specific model, because the state space is too large to explore entirely. For learning to be possible, an assumption is typically made that a PID controller exists, that is tuned to roughly track the desired trajectories.

Learning in this online setting is as follows. The torque prediction $\boldsymbol{\tau}_{ff}$ for the current desired state $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$, predicted using the current learned model $\mathbf{D}$, with the corresponding stabilizing feedback torque $\boldsymbol{\tau}_{fb}$, is applied to the robot, which results in the actual state $\mathbf{q}_a, \dot{\mathbf{q}}_a, \ddot{\mathbf{q}}_a$. The data generated in each time step is immediately used to update the parameters of the model $\mathbf{D}$. In the early stages of learning, when the learned model is still poor, the predictions are inaccurate and the system relies heavily on the feedback controller, which may not be optimally tuned. This causes the actual states to differ from the desired states, and eventually, after many trials, the model will improve and generalize to the desired states of the trajectory.

When a new robot is available to learn, it must go through the same laborious process to collect training data. Our hypothesis is that, if its model is initialized offline, with data generated by a pre-existing robot while tracking the same trajectory (knowledge gained by the old robot), it may learn to track the trajectory in fewer trials. This may reduce training time of the new robot considerably, and is beneficial particularly in cases where operating the new robot is more expensive than operating the old robot. Next we outline our proposed knowledge transfer approach to improve learning of inverse dynamics for trajectory control.

IV. Knowledge Transfer for Inverse Dynamics

We employ inter-robot transfer to improve learning to track a specified trajectory for target robot learner $\boldsymbol{\Omega}_t$ by re-using data generated from the experience of source robot learner $\boldsymbol{\Omega}_s$. To achieve this, we learn a mapping function $f : \boldsymbol{\chi}_s \mapsto \boldsymbol{\chi}_t$, for mapping samples from the domain $\boldsymbol{\chi}_s$ of the source robot to the domain $\boldsymbol{\chi}_t$ of the target robot. We assume training data $X_s \subset \boldsymbol{\chi}_s$ and $X_t \subset \boldsymbol{\chi}_t$ with correspondences, from which to learn f. This data set is assumed to be generated by the source and target robots respectively (see Section IV-A). In our case of transfer for inverse dynamics, each data sample $\boldsymbol{x}_j^i = \{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a, \ddot{\boldsymbol{q}}_a, \boldsymbol{\tau}_a\}$, where j is either s or t for source data and target data respectively, $i = 1 : n$, and n is the number of samples in X_s and X_t.

We also assume access to $\boldsymbol{\Omega}_s$'s model from which to generate source trajectory data $\boldsymbol{\xi}_s \in \Re^{m \times (4d_s)}$, where d_s is $\boldsymbol{\Omega}_s$'s DoF and m is the number of samples in the experience data. Similar to X_s, $\boldsymbol{\xi}_s$ also contains joint positions, velocities, accelerations and torques associated with the trajectory. Then we use the mapping f to transfer the source trajectory $\boldsymbol{\xi}_s$ into the domain of $\boldsymbol{\Omega}_t$ to obtain the estimated target trajectory data $\hat{\boldsymbol{\xi}}_t \in \Re^{m \times (4d_t)}$, where d_t is $\boldsymbol{\Omega}_t$'s DoF. Finally, $\hat{\boldsymbol{\xi}}_t$ is used to initialize $\boldsymbol{\Omega}_t$'s model offline, and the model is subsequently updated online as $\boldsymbol{\Omega}_t$ learns to track the desired trajectory.

Figure 2 illustrates our transfer learning-based control framework described above. We discuss the process of generating correspondences from the robots in Section IV-A and models for learning f from this sample in Section IV-B.

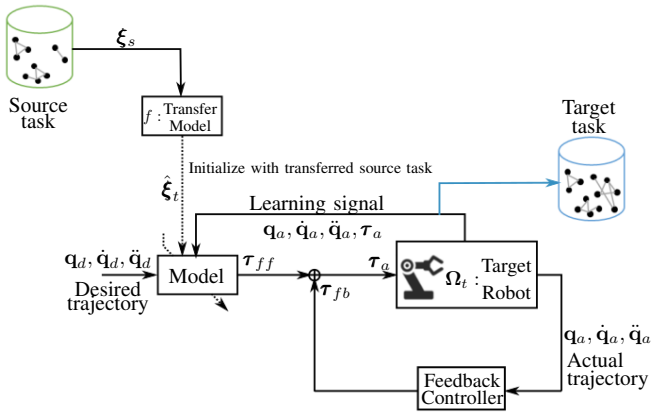


Fig. 2: Proposed transfer learning-based control framework.

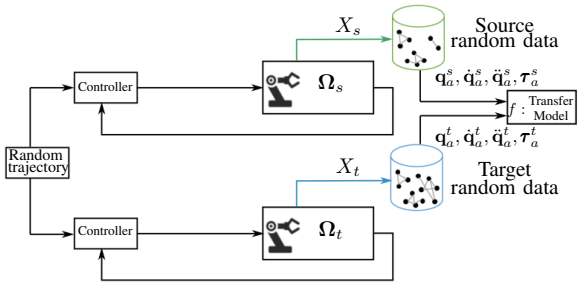


Fig. 3: Our framework for collecting correspondence data.

A. Collecting Correspondences

To train the transfer model f, we must collect correspondence data, X_s and X_t, from the robots. This means the robots must be controlled to generate similar movements, in order to identify correspondences. In [25] the authors used analytical controllers of the robots to track similar trajectories with the same speed, and used cubic spline interpolation in the joint and torque spaces to obtain correspondences. We follow the same procedure; however, we do not assume knowledge of analytical controllers, due to difficulties in accurately determining such controllers in practice as mentioned in Section I. Thus, we employ PID controllers to track similar trajectories with both robots.

We define correspondences in the task space of the robots, because that space is robot-agnostic, and also allows us to easily specify trajectories. To obtain the correspondence data in joint and torque spaces, we assume *kinematic* models of the robots are available and use them to map the trajectories into the joint space, and use a PID controller to track them. We generate random straight-line trajectories of random length in the vicinity (in task space) of the trajectories to be learned and track the same trajectories with both robots with the same speed. Thus, two data points between the robots along the straight-line trajectories are paired together as correspondences if they share the same time step.

In the general case, where the robots have non-overlapping task spaces, due to differences in kinematics, corresponding trajectories of the target robot are obtained by locating source robot trajectories in the reference frame of the target robot. This is easily achieved by aligning their task spaces, using the transformation between their specified desired trajectories[1]. This kinematic retargeting approach is widely employed in computer graphics and robotics, where a trajectory of one kinematic embodiment is projected onto the space of another kinematically different embodiment [27].

We assume both robots have PID controllers that are roughly tuned, with parameters that are not necessarily optimal. This assumption is reasonable since learning *tabula rasa* also makes the same assumption. Our experiments demonstrate that correspondence data generated in this manner is sufficient to learn transfer models and accelerate learning, where the PID parameters we used were not optimally tuned.

Fig. 3 shows our framework for collecting correspondences, where the same straight-line random trajectories have already been mapped into the joint spaces of the robots using corresponding inverse kinematics (IK) models.

B. Learning the Transfer Model

In this section we describe learning the transfer model f from a sample of correspondences generated from the robots as described in Section IV-A. We are provided with source domain data $X_s = \{\boldsymbol{x}_s^i\}_{i=1}^n$ and target domain data $X_t = \{\boldsymbol{x}_t^i\}_{i=1}^n$, where $\boldsymbol{x}_s \in \Re^{4d_s}$ and $\boldsymbol{x}_t \in \Re^{4d_t}$, and in general $d_s \neq d_t$ due to the robots potentially having differing DoFs. The aim is to learn the mapping f that we can use to transfer source domain data into the target domain, in such a way that is useful to the target robot. As previously mentioned in Section II-B, this problem can be solved using manifold alignment techniques. Manifold alignment techniques allow for knowledge transfer between two seemingly disparate data sets, by aligning their underlying manifolds [24]. This can be accomplished using two general methods: two-step alignment methods and one-step alignment methods.

In the first step of a two-step alignment method, latent representations of the source and target data are found independently in a lower dimensional space using dimensionality reduction. In the second step, a transformation between the two is computed by aligning them in the latent space. A one-step alignment method on the other hand combines the two steps into one single step, where the data sets are projected into a shared latent space. The output of this process is two mappings that map the data sets into the shared latent space. For knowledge transfer to be possible in both methods the mappings between the original space and the latent space must be bijective, as the inverses are needed to map back to the original spaces.

Examples of the two-step approach in robotics include combining Principal Component Analysis (PCA) and Procrustes Analysis (PA) [25], [30], [31]. Examples of the one-step approach include Unsupervised Manifold Alignment (UMA) [24], shared Autoencoders [32] and shared Gaussian Process Latent Variable Models [33]. All these approaches

[1]We assume the desired trajectories of both robots are already specified, as is generally the case in trajectory tracking problems.

have the property that the mappings between the spaces are guaranteed to be bijective.

Any of the approaches discussed above could be used in our framework to learn the transfer models. In our experiments the simple combination of PCA and PA proved sufficient, and enables learning from very few data points, and we compared it against the combination of PCA and the non-linear extension of PA, Local Procrustes Analysis (LPA) [30]. By applying PCA, we assume a linear relationship between the original data spaces and the corresponding latent representations. This is reasonable in our case because the training data is collected in the vicinity of the tasks to be transferred, and therefore the latent representations are expected to lie in relatively simple manifolds. Furthermore, the linear mappings of PCA are guaranteed to be bijective.

Procrustes Analysis Learning the mapping f with PCA and PA is as follows. First the data is preprocessed by subtracting the mean and whitening it, and then projected into a latent space of lower dimensionality d to obtain latent manifold representations $\boldsymbol{s} \in M_s$ and $\boldsymbol{t} \in M_t$, using $\boldsymbol{s} = B_s(\boldsymbol{x}_s - \boldsymbol{\omega}_s)$ and $\boldsymbol{t} = B_t(\boldsymbol{x}_t - \boldsymbol{\omega}_t)$. The values $\boldsymbol{\omega}_s = \mathbb{E}\{X_s\}$ and $\boldsymbol{\omega}_t = \mathbb{E}\{X_t\}$ are the means of the data, where $\mathbb{E}\{\cdot\}$ denotes the expectation operator. Matrices B_s and B_t are obtained such that the variances in X_s and X_t are maximized, respectively, and only the first d columns that maximize the variances are used to obtain the latent representations M_s and M_t.

The alignment function is then modeled as a linear mapping $f_d : M_s \mapsto M_t$ in the latent space, with $f_d(\boldsymbol{s}) = A\boldsymbol{s}$ where $A^{d\times d}$ is a transformation matrix. The expression for A was derived in [25], and is given as $A = \boldsymbol{\Sigma}_{ss}^{-1}\boldsymbol{\Sigma}_{ts}$, where $\boldsymbol{\Sigma}_{ss}$ is the covariance matrix of the source matrix M_s and $\boldsymbol{\Sigma}_{ts}$ is the covariance between the source and target matrices M_s and M_t. The reader is referred to [25] for a full derivation.

A new point $\boldsymbol{s}_\star = B_s(\boldsymbol{x}_s^\star - \boldsymbol{\omega}_s)$ in the source manifold can then be mapped to the target manifold using $\hat{\boldsymbol{x}}_t^\star = {B_t}^{\#} A\boldsymbol{s}_\star + \boldsymbol{\omega}_t$, where $\hat{\boldsymbol{x}}_t^\star$ is the transferred point and $B^{\#}$ is the Moore-Penrose inverse of B.

Local Procrustes Analysis LPA extends PA to handle non-linear mappings, by approximating a global non-linear manifold alignment with locally linear functions [30]. To achieve this, LPA first clusters the two data sets into K local clusters. Then a linear mapping for each cluster is computed using PA. A new data point from the source domain can then be mapped to the target domain by a weighted sum of the linear mappings.

In LPA, clustering is typically performed in the input space of one of the domains (the source in our experiments) using Gaussian Mixture Modeling (GMM), and the clusters are transferred to the target domain using correspondence information. Clustering in input space ensures we obtain efficient clusters, because the input and output spaces of the data sets are expected to be correlated. In our case the state of the robot $\{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a\}$ is correlated with the applied torque $\boldsymbol{\tau}_a$ through dynamics of the robot.

We slightly modify LPA in this paper in order to combine it with PCA, such that it can map between data sets of different dimensionality. Instead of applying PCA globally in the first step, as with PA, and then clustering in the latent space, we cluster the data in the state space of the original source data (i.e. $\{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a\} \in \Re^{2d_s}$) and apply PCA locally in each cluster, resulting in a PCA mixture model, capable of non-linear dimensionality reduction. We also modify the EM initialization scheme in [30], where we apply PCA and PA in each cluster. This introduces an extra parameter d which is the dimensionality of the latent space, which is kept the same for all clusters. The rest of the training procedure is the same and the reader is referred to [30] for more information about training LPA.

V. Experiments

A. Experimental Setup

We conducted experiments in simulation to transfer knowledge between the following robots: i) a 5-DoF arm of the Kuka youBot, ii) a 4-DoF Interbotix PhantomX Pincher arm (see Fig. 1), iii) a 3-DoF arm, and iv) a 2-link planar arm. All robots were simulated in V-REP[2]. The robots have different kinematic and dynamic properties[3], including different number of DoFs. The Pincher, 3-DoF and the 2-link arms are smaller, lighter and have low torque ratings on their joints – limited to 2.5 Nm for Pincher and 40 Nm for the others, whereas the youBot arm is bigger, heavier and has a relatively higher torque rating on its joints – limited to 100 Nm.

As benchmark tasks, the robots learn to track the position of two 'star-like' figures placed at different locations and orientations in the task spaces of the robots, as shown in Fig. 1. This 'star-like' trajectory has components of high acceleration, which makes tracking difficult, and is widely used as a benchmark in robot trajectory tracking control problems [34]. Each 'star' trajectory is composed of 7 straight lines starting from the center and pointing outwards.

The robots are required to follow each straight line starting from the center, going outwards and returning before following the next straight line. Each straight line is tracked for 1.2 seconds[4] in total (0.6 seconds from the center to the end and another 0.6 seconds back). Therefore one trial of tracking each full trajectory takes 8.4 seconds. The two trajectories are denoted 'Star 1' and 'Star 2' as shown in Fig. 1.

We performed several knowledge transfer experiments. Firstly, we transfer trajectories between the Pincher and the youBot arms, where we learn the transfer mappings using Procrustes Analysis and Local Procrustes Analysis, and compared this against learning from scratch and learning with a randomly initialized model. Secondly, to analyze the peformance of our knowledge transfer method between robots with very different morphologies, we transfer between the youBot, 3-DoF and 2-link arms.

We employ LWPR [10] for learning the inverse dynamics model. We tuned hyper-parameters $init\_D$ and $init\_alpha$

[2]http://www.coppeliarobotics.com/

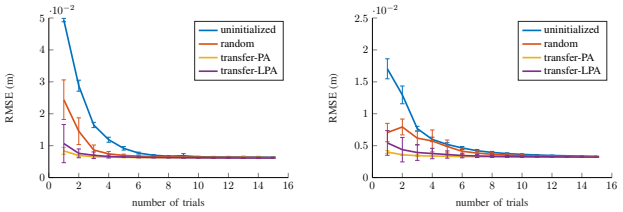
[3]We used default properties in V-REP 3.3.2 for youBot and Pincher, and the 3-DoF and 2-link planar robots were custom made.

[4]This is simulated time in V-REP.

PID gains	Pincher	youBot	3-DoF	2-link
P	5	100	5	5
I	0.01	0.01	0.01	0.01
D	0.1	2.0	0.1	0.1

LWPR params.	Pincher	youBot	3-DoF	2-link
$init\_D$	30	30	30	30
$init\_alpha$	1.1	0.01	0.001	0.001

TABLE I: Learning parameters for all robots.



(a) Transfer to youBot. (b) Transfer to Pincher.

Fig. 4: Accelerating 'Star 1' learning.

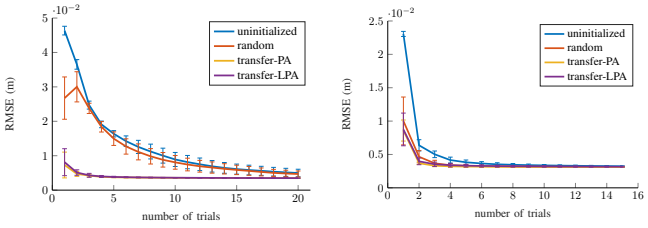
using a grid strategy, following the guideline from the LWPR software package[5], and the parameter values used are shown in Table I. The PID gains used for each robot joint are also shown in Table I. The learning procedure is as outlined in Section III. Predictions and model updates are done at 100Hz for all robots.

B. Transfer for Inverse Dynamics Tracking

The aim of this experiment is to evaluate our knowledge transfer approach and to investigate the benefit of knowledge transfer in accelerating online learning of the inverse dynamics model for trajectory tracking. The training data was generated as outlined in Section IV-A. We used random straight-line trajectories of random length and of duration 0.6 seconds each for motor babbling, resulting in the trajectories being tracked at different speeds. The robots are controlled with PID controllers to track these trajectories roughly in the vicinity of 'Star 1' and 'Star 2' trajectories, for about 7 seconds per trajectory. This means each motor babbling session lasts for about 14 seconds in total per robot, resulting in samples of correspondences with 1464 data points per robot.

The PID parameters defined in the previous section were tuned roughly such that learning to track both Star trajectories is possible, and were not necessarily optimal. In the first experiment we transfer knowledge between the youBot and the Pincher robots. The dimension of the youBot data is 20 ($\{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a, \ddot{\boldsymbol{q}}_a, \boldsymbol{\tau}_a\}$ for 5 DoFs), and that of the Pincher arm is 16 (same variables for 4 DoFs). We learned transfer models from the 1464 samples of correspondences from both robots, using PA and LPA, both combined with PCA for matching the dimensions of the data sets (see Section IV-B). We found the latent dimension $d = 16$ to be sufficient for both PA and LPA. Lower values of d lead to decreased transfer performance and values less than 10 barely transferred any useful knowledge. Figure 4 and 5 show results for initializing

[5]http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr



(a) Transfer to youBot. (b) Transfer to Pincher.

Fig. 5: Accelerating 'Star 2' learning.

the target model offline with knowledge transfer and random initialization, for 'Star 1' and 'Star 2' respectively, compared to learning from scratch (denoted 'uninitialized').

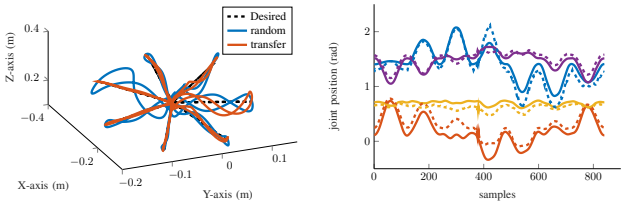
Learning from scratch Both robots successfully learn to track the trajectories from scratch over time, measured in terms of number of trials, indicated by the decreasing tracking errors. The smaller and lighter Pincher learns significantly faster and better, as it achieves lower tracking errors and converges in fewer trials (also starts with lower tracking errors). The youBot achieves slightly larger tracking errors, and converges slowly, especially for 'Star 2' where it requires more than 20 trials. The youBot arm requires larger PID gains due to its heavier components requiring larger torques to move, resulting in large feedback torques, especially in the first few trials where the learned model is still poor.

Random initialization We also separately initialize learning with random data generated in the motor babbling session as a benchmark, denoted 'random', as it has previously been shown to accelerate learning [28], [29]. This only accelerates learning in certain cases, particularly for tasks that are simple to learn. It fails to accelerate convergence of the more difficult 'Star 2' trajectory for youBot, and 'Star 1' for Pincher.

Knowledge transfer We transferred to both robots, where the robots exchange roles of being source and target. We took the data generated by the source robot when learning to track from scratch and transferred it to the target robot. This provided us with additional data (12600 points for 15 trials and 16800 for 20 trials) that we use to initialize the target robot model. We denote initializing with transfer 'transfer-PA' and 'transfer-LPA' for PA and LPA respectively.

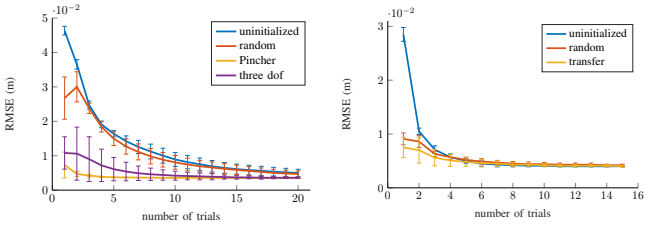
Knowledge transfer accelerates learning considerably in most cases. In particular, when the youBot is the target learning 'Star 2' (see Fig. 5a), where learning *tabula rasa* and random initialization failed to converge within 20 trials, transfer converged within 5 trials. Both transfer models perform well, with PA slightly better than LPA in the early trials of 'Star 1'. This indicates that linear mappings are sufficient to transfer useful knowledge in this case.

Figure 6a shows example end-effector trajectories for random initialization and transfer in the first learning trial of the youBot learning 'Star 2'. We observe that for the most part transfer leads to stable and safe learning already in the first trial. This is due to transfer biasing exploration into relevant spaces near the desired trajectory, thus resulting



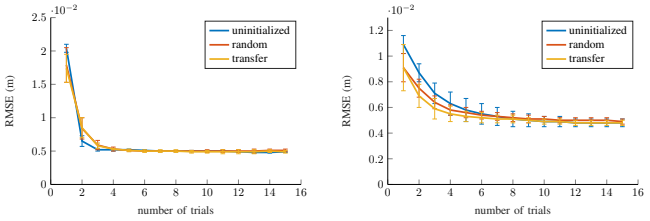
(a) First trial for 'Star 2'. (b) First 4 joints of youBot.

Fig. 6: Example transfer results for youBot.



(a) Transfer to youBot. (b) Transfer to 3-DoF.

Fig. 7: youBot and 3-DoF transfer.



(a) Transfer to youBot. (b) Transfer to 2-DoF.

Fig. 8: Transfer with a 2-link robot.

in more efficient exploration.

In Fig. 6b, we show an example of mapping the joints of the Pincher into the youBot domain using PA, where the solid lines are youBot joints and the dashed lines are transferred Pincher joints. We can see that although the linearly transferred Pincher joints are not exactly aligned with the target joints, they are distributed similarly, making it possible for learning to be accelerated.

Figure 7 shows results for the second experiment, where we transfer knowledge between the youBot and the 3-DoF robot. The setup is the same as the previous experiment. For brevity we only show results for the difficult 'Star 2' trajectory, however the results also generalize to 'Star 1'. The dimensionality of the 3-DoF arm data is 12, and the latent dimension $d = 7$ was used. The 3-DoF robot is also able to learn to track the trajectories faster than the youBot as it has lighter components, and also because it has lower dimensional state and action spaces. Random initialization is also able to accelerate convergence in this case, because of the lower dimensionality of the spaces, which makes learning easier.

In Fig. 7a we observe that knowledge transfer from the much lower dimensional, and also kinematically very different, 3-DoF arm to the higher dimensional youBot arm is less efficient compared to transfer from the 4-DoF Pincher arm, as the lower dimensional arm has fewer DoFs compared to the redundant higher dimensional arm. However, transfer is still able to accelerate convergence of the higher DoF youBot arm compared to random initialization and learning from scratch. Transfer to the lower dimensional 3-DoF arm from the youBot arm also accelerates learning, although the benefit is not as much as it is already simpler for the lower dimensional arm to learn.

Figure 8 shows results for the last experiment, where we transfer between the youBot and the 2-link arm. In this experiment, the trajectories are in a 2D plane, as the task space of the 2-link robot is 2D, and thus is incapable of transferring any useful knowledge to the youBot arm for trajectories in 3D task spaces. We also sped up the tasks to 0.5 seconds per straight line trajectory, making it harder for the 2-link robot to learn. The dimensionality of the 2-link data is 8 and the latent dimension $d = 5$ was used.

Transferring to the youBot from a simple 2-link is not effective (see Fig. 8a), as the initial jump in learning is small, however learning for the youBot in 2D is already too easy for transfer to be required. On the other hand, transfer from the youBot to the 2-link (see Fig. 8b) accelerates learning, although learning for the simple 2-link robot is already easy.

In the general case of robots learning to track multiple trajectories, transfer saves more time because the robots need only to generate motor babbling data once in the beginning. Table II shows results demonstrating the benefit of transfer between the Pincher and youBot, where (Value1,Value2) indicate number of trials to converge for 'Star 1' and 'Star 2' respectively, and the values in brackets indicate the total time for learning to track both trajectories. Note that for *random* and *transfer* the total time includes 14 seconds for motor babbling.

For instance, in the case of transferring to the youBot, learning from scratch for 'Star 1' converges within 9 trials (75.6 seconds, 8.4 seconds per trial) and with knowledge transfer it converges within 4 trials (33.6 seconds). Combining with the results of 'Star 2', 20 trials (168 seconds) for learning from scratch and 5 trials (42 seconds) with transfer, and considering the 14 seconds for motor babbling, knowledge transfer saves up to 63.2% of training time in total, whereas random initialization only saves up to 4.6%. For the Pincher, the benefit of transfer is slightly less, at 58.7%, while that of random initialization is 15.8%. This is due to the fact that learning for the Pincher is simpler. Note, however, that the benefit of knowledge transfer would increase when more trajectories must be learned in the same task space, since transfer models need only to be learned once.

VI. CONCLUSIONS

This paper proposed a knowledge transfer scheme for accelerating online learning of the inverse dynamics model for trajectory tracking, where PID controllers were used to generate training data from the robots for learning knowledge transfer models. We demonstrated the benefit of transfer

Setting	Pincher [time (s)]	youBot [time (s)]
Scratch	(12,9) [176.4]	(9,20) [243.6]
Random Accel.	(12,4) [148.4] 15.8%	(6,20) [232.4] 4.6%
Transfer Accel.	(3,4) [72.8] **58.7%**	(4,5) [89.6] **63.2%**

TABLE II: Analysis of the benefit of knowledge transfer. *Accel.* represents the percent gain by accelerating learning using the corresponding method. (Value1,Value2) indicate number of trials to converge for 'Star 1' and 'Star 2' respectively, and [time] the total time for both.

on several robots with different kinematic and dynamic properties, and showed that transfer biases exploration into relevant spaces, which leads to accelerated learning and safe exploration from the start to the end of the learning process. We also demonstrated that our approach is capable of learning transfer models between arms with very different kinematic structures, as shown by learning transfer models between a 5-DoF redundant arm and 2-link arm.

Since our experiments were conducted in simulation, future work will look into validating our results on real robots, as physics based simulators are not guaranteed to represent all complex dynamics of the real world. Also, we will look into employing several other model learning techniques reviewed in Section II-A, as this will ensure that the conclusions drawn are not biased by a specific model learning technique.

ACKNOWLEDGMENT

This research was supported by Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST) and SOINN Inc.

REFERENCES

[1] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, 'Learning Inverse Dynamics : a Comparison,' European Sym. on Artificial Neural Networks, 2008, pp. 13–18.

[2] R. Featherstone, 'Rigid Body Dynamics Algorithms,' New York: Springer, 2008, pp. 39–64.

[3] D. Nguyen-Tuong, and J. Peters, 'Model learning for robot control: A survey,' Cognitive Processing, vol. 12, no. 2, pp. 319–340, 2011.

[4] A. S. Polydoros, L. Nalpantidis, and V. Kruger, 'Real-time deep learning of robotic manipulator inverse dynamics,' IEEE Int. Conf. on Intelligent Robots and Systems, 2015, pp. 3442–3448.

[5] A. Gijsberts, and G. Metta, 'Incremental learning of robot dynamics using random features,' IEEE Int. Conf. on Robotics and Automation, 2011, pp. 951–956.

[6] D. Nguyen-Tuong, B. Scholkopf, and J. Peters, 'Sparse online model learning for robot control with support vector regression,' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2009, pp. 3121–3126.

[7] D. Nguyen-Tuong, and J. Peters, 'Incremental Sparsification for Real-time Online Model Learning,' in Artificial Intelligence and Statistics 2010, pp. 557–564.

[8] A. Gijsberts, and G. Metta, 'Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression,' Neural Networks, vol. 41, pp. 59–69, 2013.

[9] C. G. Atkeson, A. W. Moore, and S. Schaal, 'Locally Weighted Learning for Control,' Artificial Intelligence Review, vol. 11, pp. 75–113, 1997.

[10] S. Vijayakumar, A. D'Souza, and S. Schaal, 'Incremental online learning in high dimensions,' Neural Computation, vol. 17, no. 12, pp. 2602–2634, 2005.

[11] D. Nguyen-Tuong, M. Seeger, and J. Peters, 'Model Learning with Local Gaussian Process Regression,' Advanced Robotics, vol. 23, no. 15, pp. 2015–2034, 2009.

[12] F. Meier, P. Hennig, and S. Schaal, 'Incremental Local Gaussian Regression,' Advances in Neural Information Processing Systems, vol. 27, pp. 972–980, 2014.

[13] Y. Choi, S. Cheong, and N. Schweighofer, 'Local Online Support Vector Regression for Learning Control,' Int. Sym. on Computational Intelligence in Robotics and Automation, 2007, pp. 13–18.

[14] F. Meier, and S. Schaal, 'Drifting Gaussian Processes with Varying Neighborhood Sizes for Online Model Learning,' IEEE Int. Conf. on Robotics and Automation, 2016.

[15] D. Nguyen-Tuong, and J. Peters, 'Using model knowledge for learning inverse dynamics,' IEEE Int. Conf. on Robotics and Automation, 2010, pp. 2677–2682.

[16] D. Romeres, M. Zorzi, R. Camoriano, and A. Chiuso, 'Online semi-parametric learning for inverse dynamics modeling,' IEEE Conf. on Decision and Control, 2016, pp. 2945–2950.

[17] J. S. de la Cruz, D. Kulić, and W. Owen, 'Online Incremental Learning of Inverse Dynamics Incorporating Prior Knowledge', Int. Conf. of Autonomous and Intelligent Systems, 2011, pp. 167–176.

[18] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, 'Incremental semiparametric inverse dynamics learning,' IEEE Int. Conf. on Robotics and Automation, 2016, pp. 544–550.

[19] F. Meier, D. Kappler, N. Ratliff, and S. Schaal, 'Towards Robust Online Inverse Dynamics Learning,' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2016, pp. 4034–4039.

[20] S. J. Pan, and Q. Yang, 'A Survey on Transfer Learning,' in IEEE Trans. on Knowledge and Data Engineering, vol. 10, no. 20, pp. 1345-1359, 2010.

[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 e-prints, 2016.

[22] G. Chowdhary, T. Wu, M. Cutler and J. P. How, 'Rapid transfer of controllers between UAVs using learning-based adaptive control,' IEEE Int. Con. on Robotics and Automation, 2013, pp. 5409-5416.

[23] J. C. G. Higuera, D. Meger and G. Dudek, 'Adapting learned robotics behaviours through policy adjustment,' IEEE Int. Conf. on Robotics and Automation, 2017, pp. 5837-5843.

[24] C. Wang, and S. Mahadevan, 'Manifold alignment without correspondence,' Int. Joint Conf. on Artificial Intelligence, 2009, pp. 1273–1278.

[25] B. Bócsi, L. Csató, and J. Peters, 'Alignment-based transfer learning for robot models,' Int. Joint Conf. on Neural Networks, 2013, pp. 1-7.

[26] M. K. Helwa, and A. Schoellig, 'Multi-Robot Transfer Learning: A Dynamical System Perspective,' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2017, accepted.

[27] G. Maeda, M. Ewerton, D. Koert, and J. Peters, 'Acquiring and Generalizing the Embodiment Mapping from Human Observations to Robot Skills,' IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 784-791, 2016.

[28] J. S. de la Cruz, D. Kulić, and W. Owen, 'A Comparison of Classical and Learning Controllers,' 18th IFAC World Congress, 2011, pp. 1102–1107.

[29] J. Peters, and S. Schaal, 'Learning to Control in Operational Space,' The Int. J. of Robotics Research, vol. 27, no. 2, pp. 197–212, 2008.

[30] N. Makondo, B. Rosman, and O. Hasegawa, 'Knowledge transfer for learning robot models via Local Procrustes Analysis,' IEEE-RAS Int. Conf. on Humanoid Robots, 2015, pp. 1075-1082.

[31] M. Hiratsuka, N. Makondo, B. Rosman, and O. Hasegawa, 'Trajectory learning from human demonstrations via manifold mapping,' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2016, pp. 3935-3940.

[32] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, 'Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning,' IEEE Int. Conf. on Learning Representations, 2017, in Print.

[33] B. Delhaisse, D. Esteban, L. Rozo, and D. Caldwell, 'Transfer learning of shared latent spaces between robots with similar kinematic structure,' Int. Joint Conf. on Neural Networks, 2017, pp. 4142-4149.

[34] J. S. de la Cruz, W. Owen and D. Kulc, 'Online learning of inverse dynamics via Gaussian Process Regression,' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2012, pp. 3583-3590.