Alignment-based Transfer Learning for Robot Models

Botond Bócsi Lehel Csató Jan Peters

***Abstract*— Robot manipulation tasks require on robot models. When exact physical parameters of the robot are not available, learning robot models from data becomes an appealing alternative. Most learning approaches are formulated in a supervised learning framework and are based on clearly defined training sets. We propose a method that improves the learning process by using additional data obtained from other experiments of the robot or even from experiments with different robot architectures. Incorporating experiences from other experiments requires *transfer learning* that has been used with success in machine learning. The proposed method can be used for arbitrary robot model, together with any type of learning algorithm. Experimental results indicate that task transfer between different robot architectures is a sound concept. Furthermore, clear improvement is gained on forward kinematics model learning in a task-space control task.**

I. Introduction

Robot manipulation tasks ofter require learning robot models. These models incorporate our knowledge about the architecture of the robot. When the physical parameters of the robot are unknown or inaccurate, analytical control methods fail, since the robot model is unavailable. In such cases, learning the robot model from data is an appealing alternative as it requires only samples gained from experiments. Collecting data from experiments is considered to be significantly easier than obtaining an accurate analytical physical model of the robot [1]. Inferring robot models form data has a broad literature [1]–[7]. Most model learning problems are formulated in standard supervised learning framework, where the training data consists of inputs associated with labels. This formulation leads to algorithms where the learning process is based on unequivocally defined training sets, acquired either online or offline. We propose an approach where additional datasets can be used.

The motivation of this paper comes from human learning. A fundamental difference between human and machine learning is that robots have no prior knowledge of the world, whereas humans have at their disposal past experiences, e.g., even though the new task has not yet been performed by the human learner, previous life-experience would certainly speed up learning. For example, when one has to learn how to play table tennis, the fact that he uses his hand every day improves the learning process. Furthermore, if he has done *similar* activities before, such as tennis, the improvement will be more significant.

Botond Bócsi and Lehel Csató are with Faculty of Mathematics and Informatics, Babeş-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania, {bboti, lehel.csato}@cs.ubbcluj.ro

Jan Peters is with Technische Universitaet Darmstadt, Intelligent Autonomous Systems Group, Hochschulstr. 10, 64289 Darmstadt, Germany, mail@jan-peters.net

Jan Peters is with Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Spemannstraße 38, 72076 Tübingen, Germany

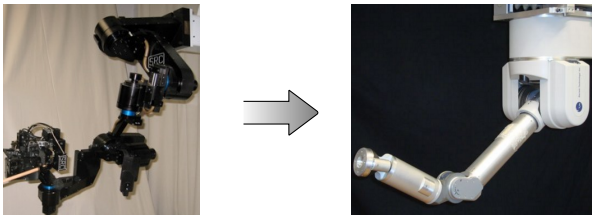


Fig. 1: Transfer learning between a Sarcos Master arm (left) and a Barrett WAM (right).

In this paper, we consider the scenario when the robot uses its own – or other – past experiences to improve its learning speed. For example, consider a humanoid robot with two arms which learned a given task with one of its arms. We propose a method that helps to learn a similar task with the other hand based on the knowledge gained from the first task.

The idea of transferring knowledge between different tasks is not novel, it has been considered in machine learning under the name of *transfer learning* [8]–[10]. Transfer learning is based on the insight that the learning process of a given task can be improved when some *knowledge* gained from other learning problems is reused. In robotics, it has been mainly applied to improve reinforcement learning tasks as part of lifelong learning [11] – the general idea that any learning process must be based on information gained from previously learned tasks. The method proposed in this paper improves robot model learning using transfer learning techniques.

The paper is organized as follows: a general introduction to transfer learning is given in Section I-A, also highlighting the existing robot learning applications. Then, we present the prevailing robot model learning approaches we aim to improve in Section I-B. Section II details the proposed transfer learning algorithm, while in Section III we show applications where our method improves robot model learning. Conclusions and future work are discussed in Section IV.

A. Transfer Learning

The concept of *transfer learning* does not have a unique definition, and it has been investigated in different scenarios. The common feature of all scenarios is that the learning process is improved by taking additional data beside the training dataset [8]. In every transfer learning framework a source task and a target task are associated with the source and the target domain respectively. The typical procedure has two steps: first, the source task is learned, then the

"knowledge" from the first step is used to improve the target task. We mention different transfer learning paradigms.

Inductive transfer learning [12] solves problems where source and target domains coincide but the tasks are different. In transductive transfer learning [9], [13] the source domain and the target domain are different but the same task has to be solved for both cases.

We classify transfer learning methods based on the principles with which they transfer knowledge between the tasks. First, they may transfer knowledge of instances [12] where parts of the source task set is reused. Second, they may transfer knowledge of feature representation [9] when a better task feature representation is found based on the source task. Third, they may transfer knowledge of parameters [5], [13], [14] when the estimated parameters of the source model are used. The main application of the third approach is in a Bayesian setting, here, the values of the source parameters can be reused in a natural way by setting them as prior parameters of the target model [5], [13]–[15].

Transfer learning methods have been applied with success in text categorization [15], [16], boosting [12], naive Bayes classification [13], and breast cancer detection [16].

Our approach has similarities to the one of Pan et al. [9]. They use dimensionality reduction to find a common latent feature space of the source data and task data. Standard supervised learning algorithms are applied with the latent feature space representation of the data as inputs and the true labels as outputs. Their application shows performance improvements in a wifi localization task (measuring the strength of wifi signals, when one aims to locate itself in a building [9]) and binary text classification.

To best of our knowledge there were a few attempts to apply the transfer learning framework in robot learning. Most of the approaches used transfer learning in reinforcement learning tasks. Knowledge transfer was achieved by reusing instances of data, action-value functions, policies, or full task models [10], [11]. An early attempt to use transfer learning in robot model learning is in multi-task learning, where multiple Gaussian processes have been applied to model inverse dynamics of a robot arm [5]. To do so, multiple Gaussian process instances shared the hyper-parameters of the model.

We propose transfer learning based learning algorithm that can be applied with several of robot model learning methods since we do not define a novel learning method but a data transfer mechanism. Furthermore, the proposed method can be used for learning any type of robot model, e.g. forward kinematics, inverse kinematics, inverse dynamics. In what follows, we provide a short overview of different robot model learning methods.

B. Learning Robot Models

In robot model learning the approximation of different robot models is addressed. We enumerate forward kinematics, inverse kinematics, inverse dynamics, or operational space control. A survey on robot model learning is presented in Nguyen-Tuong and Peters [1].

Forward kinematics is the simplest to approximate from the aforementioned models. It is a one-to-one function, mapping joint coordinates to end-effector positions, i.e., $\boldsymbol{\theta} \rightarrow \mathbf{x}$, where $\boldsymbol{\theta} \in \Re^{\mathfrak{n}}$ and $\mathbf{x} \in \Re^{\mathfrak{p}}$ when the robot has $\mathfrak{n}$ degrees of freedom (DoFs) and the operational space (or task-space) is $\mathfrak{p}$ dimensional. Learning forward kinematics can be treated as an ordinary regression problem, and standard methods, such as neural networks [17] and locally weighted projection regression (LWPR) [7] were used with success to model it.

A second model is the inverse dynamics model. It expresses the torque that should be applied to obtain a desired joint configuration. It is again a locally unique mapping from the joint-space to the torque-space, i.e., $\boldsymbol{\theta} \rightarrow \boldsymbol{\tau}$. The torque $\boldsymbol{\tau}$ belongs to an $\mathfrak{n}$ dimensional space, $\boldsymbol{\tau} \in \Re^{\mathfrak{n}}$, since the model associates a torque value to every joint. Standard regression techniques, such as LWPR [2], support vector regression [2], Gaussian process regression [2], [5], neural networks [3], all result in accurate models.

A third model, the inverse kinematics, performs a mapping from end-effector positions $\mathbf{x}$ to joint coordinates $\boldsymbol{\theta}$. This mapping may not be a one-to-one function when a given end-effector position can be reached from more than a single joint configuration. Due to the multiple outputs, inverse kinematics is an ill-defined function; thus, standard learning methods fail. To overcome this difficulty, LWPR [6], structured output learning [4], and paired neural networks have been used [17].

The fourth family of robot models, operational space control [18], couples inverse kinematics and inverse dynamics by directly modeling the transformation between the desired end-effector position and the torque that should be applied to reach the positions, i.e., $\mathbf{x} \rightarrow \boldsymbol{\tau}$. Learning operational space control is a novel approach in robot control. Most of the approaches are kernel-based methods and take advantage of the flexibility and the good generalization property of these methods. Gaussian processes [19] and LWPR [20] have also been used in this context.

The common feature of the above robot model learning problems is that they need samples. Acquiring these samples can be difficult, time consuming, or very costly. We propose a method that helps obtaining samples from past experiences of the robot, or even from experiences of different robot architectures. Next, we present how to model data transformation from other experiments such that it becomes useful in a new learning task.

II. Knowledge Transfer in Robot Learning

We aim to improve the learning process for the previous robot models based on information transfer from past experiments in the form of additional datasets. In the remainder of this paper, for simplicity, knowledge transfer in robot model learning is presented for forward kinematics. However, we emphasize that our algorithm is not limited to these models and could equally be applied to all categories from above. For example, in our experiments we use it for inverse dynamics learning.

Regardless of the learning framework, a source task – a task that has already been learned – and a target task – a

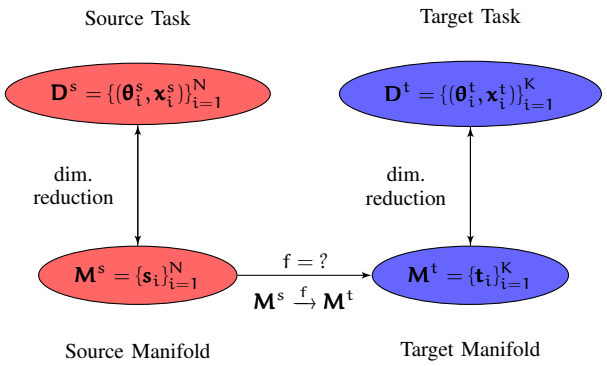


Fig. 2: Transfer learning scheme: first, we obtain a common low dimensional manifold of the source dataset $\mathbf{D}^s$ and target dataset $\mathbf{D}^t$, i.e., $\dim(\mathbf{M}^s) = \dim(\mathbf{M}^t)$, using dimensionality reduction; then, we find a mapping $f()$ between these manifolds.

problem that is difficult to learn for some reasons – has to be defined. For example, it may happen that we have a limited time access to the target architecture or we want to do as few operations on it as possible.

We consider the case when a source task has a dataset $\mathbf{D}^s = \{(\boldsymbol{\theta}_i^s, \mathbf{x}_i^s)\}_{i=1}^N$, and we aim to improve the learning of a target task that has a training set $\mathbf{D}^t = \{(\boldsymbol{\theta}_i^t, \mathbf{x}_i^t)\}_{i=1}^K$. Since we consider forward kinematics, we assume the inputs are joint configurations and the labels are end-effector positions. The proposed method can be applied in two scenarios: first, when K is smaller than N, thus, we do not have enough training data for the target task to obtain an accurate robot model; second, when it is easier to perform the source task and then transform it into the target task. We emphasize that it is not required that the dimension of the data from $\mathbf{D}^s$ to be same as the dimension of the data from $\mathbf{D}^t$. Both the joint-space dimension and the end-effector space dimension can be different, i.e., $\dim(\boldsymbol{\theta}^s) \neq \dim(\boldsymbol{\theta}^t)$, $\dim(\mathbf{x}^s) \neq \dim(\mathbf{x}^t)$.

Our algorithm is divided into two steps: first, we find a low dimensional representation of the datasets $\mathbf{D}^s$ and $\mathbf{D}^t$, with the same dimensionality, denoted with $\mathbf{M}^s$ and $\mathbf{M}^t$, i.e., $\dim(\mathbf{M}^s) = \dim(\mathbf{M}^t)$. As a second step, we find a transformation between the low dimensional manifolds, i.e., $f : \mathbf{M}^s \rightarrow \mathbf{M}^t$. The scheme of the transfer learning algorithm is presented on Figure 2. We discuss the calculation of the transformation function $f(\cdot)$ in Section II-A. In the next two sections we present the two steps of the algorithm in detail.

A. Dimensionality Reduction

Dimensionality reduction aims to find a lower dimensional representation of the data such that properties of interest are preserved, e.g., distance between the data points or maximum variance [21]. In our framework, we do not put any restrictions on the selected preserved property. However, the method must provide a bijective mapping between the low and high dimensional spaces, since the inverse mapping is also needed (see Algorithm 1 for details).

In our experiments, we used principal component analysis (PCA) as the dimensionality reduction method, since our kinematics data lay on a relatively simple manifold. By applying PCA, we assume linear relationship between the low and high dimensional manifolds. First, we center the data, that is, subtract the mean, then whiten it, that is, divide with the standard variance. The projection looks as follows

$$\begin{aligned} \mathbf{s} &= \mathbf{B}_s(\mathbf{d}^s - \boldsymbol{\mu}_s) \\ \mathbf{t} &= \mathbf{B}_t(\mathbf{d}^t - \boldsymbol{\mu}_t), \end{aligned}$$

where $\mathbf{s} \in \mathbf{M}^s$, $\mathbf{t} \in \mathbf{M}^t$, $\mathbf{d}^s \in \mathbf{D}^s$, and $\mathbf{d}^t \in \mathbf{D}^t$. The values $\boldsymbol{\mu}_s = E\{\mathbf{D}^s\}$ and $\boldsymbol{\mu}_t = E\{\mathbf{D}^t\}$ are the means of the original data, where $E\{\cdot\}$ denotes the expectation operator. Matrices $\mathbf{B}_s$ and $\mathbf{B}_t$ are transformation matrices obtained such that the variances of $\mathbf{M}^s$ and $\mathbf{M}^t$ are maximized, for details consult Lee and Verleysen [21]. Note that the bijective requirement of the dimensionality reduction method is fulfilled since the inversion of the projection function is straightforward by calculating $\mathbf{B}_s^{-1}$ and $\mathbf{B}_t^{-1}$ respectively.

An important question is how to set the common dimensions of the manifolds. From a theoretical point of view one would argue that bigger number of dimensions would lead to less information loss caused by the dimensionality reduction. Although this assumption is true in general, experiments conducted on data originated from robot model learning tasks show that other properties of the transformation also has to be taken into account, such as stability or smoothness. In our experiments, we give some general principals how to chose the dimension of the manifolds, see Section III-A for details.

B. Manifold Alignment

We model the manifold alignment function as a linear mapping $f : \mathbf{M}^s \rightarrow \mathbf{M}^t$, with

$$f(\mathbf{s}) = \mathbf{A}\mathbf{s}, \tag{1}$$

where $\mathbf{A} \in \Re^{J \times J}$ is a transformation matrix with J as the dimension of the manifolds. The definition of the mapping $f(\cdot)$ is discussed in two cases. In the first case direct correspondence between the data points is known. By direct correspondence we mean that $\mathbf{D}^s$ and $\mathbf{D}^t$ contain the same number of points and the points come in pairs. This setup can be used, for example, when the same task has been performed both in the source space and in the target space. A solution was provided by Wang and Mahadevan [22] who assumed that $f(\cdot)$ is a linear function and minimized $\|\mathbf{T} - f(\mathbf{S})\|_F$ where $\mathbf{S}$ and $\mathbf{T}$ are matrices formed from the data of $\mathbf{M}^s$ and $\mathbf{M}^t$, and $\|\cdot\|_F$ is the Frobenius norm. In this paper, we assume a linear function and minimize the expected loss of the transformation.

In the second case, there is no direct correspondence between the points of $\mathbf{D}^s$ and $\mathbf{D}^t$. They may even contain different number of points, i.e., $|\mathbf{D}^s| \neq |\mathbf{D}^t|$ as discussed by Pan et al. [9] who minimized the distance between the distributions of the datasets based on maximum mean discrepancy. Similarly to their work, we assume Gaussian distributions and find a transformation that minimizes the KL distance.

Algorithm 1 Transfer learning for robot models with PCA.

IN: $\mathbf{D}^s = \{\mathbf{d}_i^s = (\boldsymbol{\theta}_i^s, \mathbf{x}_i^s)\}_{i=1}^N$, $\mathbf{D}^t = \{\mathbf{d}_i^t = (\boldsymbol{\theta}_i^t, \mathbf{x}_i^t)\}_{i=1}^K$

{Obtain $\mathbf{M}^s = \{\mathbf{s}_i\}_{i=1}^N$ and $\mathbf{M}^t = \{\mathbf{t}_i\}_{i=1}^K$ such as}
$\mathbf{s}_i = \mathbf{B}_s(\mathbf{d}_i^s - \boldsymbol{\mu}_s)$, $\forall \mathbf{d}_i^s \in \mathbf{D}^s$ {see Section II-A}
$\mathbf{t}_i = \mathbf{B}_t(\mathbf{d}_i^t - \boldsymbol{\mu}_t)$, $\forall \mathbf{d}_i^t \in \mathbf{D}^t$ {see Section II-A}

$\mathbf{A} = \boldsymbol{\Sigma}_{ss}^{-1}\boldsymbol{\Sigma}_{ts}$ {with direct correspondence}
{OR}
$\mathbf{A} = \mathbf{U}_t\boldsymbol{\Lambda}_t^{1/2}\boldsymbol{\Lambda}_s^{-1/2}\mathbf{U}_s^\top$ {with rough alignment}

OUT: $\mathbf{B}_t^{-1}\mathbf{A}\mathbf{S} + \boldsymbol{\mu}_t$ {$\mathbf{S}$ is a matrix of all $\mathbf{s}_i \in \mathbf{M}^s$}

1) Alignment with Direct Correspondence: When direct correspondence between the datasets is known, we are given $\mathbf{M}^s = \{\mathbf{s}_i\}_{i=1}^N$ and $\mathbf{M}^t = \{\mathbf{t}_i\}_{i=1}^K$, with $N = K$, and we also know that $\mathbf{s}_i$ matches $\mathbf{t}_i$ for every $i = \overline{1, N}$. We want to find the parameter $\mathbf{A}$ from Equation (1) such that the expectation of the error of the transformation $L(\mathbf{A})$ is minimized, i.e.,

$$\mathbf{A} = \arg\min_{\mathbf{A}} L(\mathbf{A}),$$

with

$$\begin{aligned} L(\mathbf{A}) &= \mathbf{E}\left\{(\mathbf{t} - \mathbf{A}\mathbf{s})^\top(\mathbf{t} - \mathbf{A}\mathbf{s})\right\} \\ &= \mathbf{E}\left\{\mathbf{t}^\top\mathbf{t} - 2\mathbf{t}^\top\mathbf{A}\mathbf{s} + \mathbf{s}^\top\mathbf{A}^\top\mathbf{A}\mathbf{s}\right\} \\ &= \mathbf{E}\left\{\mathrm{tr}(\mathbf{t}\mathbf{t}^\top) - 2\mathrm{tr}(\mathbf{A}\mathbf{s}\mathbf{t}^\top) + \mathrm{tr}(\mathbf{A}\mathbf{s}\mathbf{s}^\top\mathbf{A}^\top)\right\} \\ &= \mathrm{tr}\left(\boldsymbol{\Sigma}_{tt} - 2\mathbf{A}^\top\boldsymbol{\Sigma}_{ts} + \mathbf{A}^\top\boldsymbol{\Sigma}_{ss}\mathbf{A}\right), \end{aligned} \quad (2)$$

where $\boldsymbol{\Sigma}_{ss}$, $\boldsymbol{\Sigma}_{tt}$, and $\boldsymbol{\Sigma}_{ts}$ are covariance matrices. The minimization can be performed by setting the derivative of $L(\mathbf{A})$ from Equation (2) to zero. After differentiation, we get

$$\begin{aligned} 0 &= -2\boldsymbol{\Sigma}_{ts} + 2\mathbf{A}^\top\boldsymbol{\Sigma}_{ss}, \\ \mathbf{A} &= \boldsymbol{\Sigma}_{ss}^{-1}\boldsymbol{\Sigma}_{ts}. \end{aligned} \quad (3)$$

The bottleneck in the computation of $\mathbf{A}$ is the inversion of the covariance matrix $\boldsymbol{\Sigma}_{ss}$. As long as the number of the DoFs of the robot is small, the computation of $\mathbf{A}$ can be done efficiently [23]. Since $\boldsymbol{\Sigma}_{ss}$ is a full-rank matrix (we assume strictly positive definiteness, thus, full-rank), the inversion is well defined. The pseudo-code of the algorithm is presented in Algorithm 1.

2) Rough Alignment: Rough alignment is also appealing since in most cases we do not have points that are paired. The distance between the distributions defined by the points of $\mathbf{M}^s$ and $\mathbf{M}^t$ is minimized. The task is now to match the two distributions defined on manifolds $\mathbf{M}^s$ and $\mathbf{M}^t$ obtained from the dimensionality reduction step. In the following, we also assume that both datasets are Gaussian distributed and we minimize the distance between the two Gaussians $p(\mathbf{M}^s)$ and $p(\mathbf{M}^t)$. Defined as the Kullback-Leibler divergence [24], the divergence has an analytical form when the distributions are Gaussian:

$$\begin{aligned} 2\mathrm{KL}\left(p(\mathbf{M}^s)\|p(\mathbf{M}^t)\right) = (\boldsymbol{\mu}_s - \boldsymbol{\mu}_t)^\top\boldsymbol{\Sigma}_{tt}^{-1}(\boldsymbol{\mu}_s - \boldsymbol{\mu}_t) + \\ + \mathrm{tr}\left(\boldsymbol{\Sigma}_{ss}\boldsymbol{\Sigma}_{tt}^{-1} - \mathbf{I}\right) - \ln\left|\boldsymbol{\Sigma}_{ss}\boldsymbol{\Sigma}_{tt}^{-1}\right|, \end{aligned} \quad (4)$$

where $\boldsymbol{\mu}_s = \mathbf{E}\{\mathbf{s}\}$ and $\boldsymbol{\mu}_t = \mathbf{E}\{\mathbf{t}\}$ are the means of $\mathbf{M}^s$ and $\mathbf{M}^t$. We use the linearity assumption presented in Equation (1), $\mathbf{M}^t = \mathbf{A}\mathbf{M}^s$. After centering the data, i.e., subtracting the mean from every data point, $\boldsymbol{\mu}_s = \boldsymbol{\mu}_t = \mathbf{0}$, the first term from the expression in (4) vanishes. The expression from Equation (4) achieves its minimum zero, when both the second term and the third term is equal to zero, i.e., when $\boldsymbol{\Sigma}_{ss}\boldsymbol{\Sigma}_{tt}^{-1} = \mathbf{I}$, leading to

$$\boldsymbol{\Sigma}_{tt} = \mathbf{A}\boldsymbol{\Sigma}_{ss}\mathbf{A}^\top. \quad (5)$$

This expression is quadratic in $\mathbf{A}$ and does not have a unique solution. To see the non-uniqueness but, nonetheless, obtain a constructive solution, we apply the eigenvalue decomposition of the covariance matrices. After the decomposition, Equation (5) looks as follows

$$\mathbf{U}_t\boldsymbol{\Lambda}_t\mathbf{U}_t^\top = \mathbf{A}\mathbf{U}_s\boldsymbol{\Lambda}_s\mathbf{U}_s^\top\mathbf{A}^\top, \quad (6)$$

where $\mathbf{U}_s$ and $\mathbf{U}_t$ are rotation matrices, i.e., $\mathbf{U}_s\mathbf{U}_s^\top = \mathbf{I}$, and $\boldsymbol{\Lambda}_s$ and $\boldsymbol{\Lambda}_t$ are diagonal matrices with the eigenvalues of $\boldsymbol{\Sigma}_{ss}$ and $\boldsymbol{\Sigma}_{tt}$, respectively. Equation (6) reveals that there are no constrains on how the dimensions of the two manifolds correspond to each other. The matrix $\mathbf{A}$ may contain any permutation of the rows, thus, we can construct solutions where we explicitly define the correspondence of the dimensions. So far, we did not assume anything about how the dimensions of the Gaussian variable correspond to each other. We assume that they correspond according to the increasing order of their variance (intuitively, in practice it means that we map joint to each other with similar variance). This constraint is achieved by arranging the columns of $\boldsymbol{\Lambda}_s$, $\mathbf{U}_s$ and $\boldsymbol{\Lambda}_t$, $\mathbf{U}_t$ such that the eigenvalues are in an increasing order. One can see that an $\mathbf{A}$ that satisfies Equation (6) has the following form

$$\mathbf{A} = \mathbf{U}_t\boldsymbol{\Lambda}_t^{1/2}\boldsymbol{\Lambda}_s^{-1/2}\mathbf{U}_s^\top. \quad (7)$$

Note that since in our experiments we used PCA in the dimension reduction step, the data are already aligned along the direction of their variance. Thus, the rotational matrices $\mathbf{U}_s$ and $\mathbf{U}_t$ from this step do not do any rotation and contain zeros outside their diagonals.

The computational complexity of $\mathbf{A}$ is the same as the eigenvalue decompositions of the covariance matrices since the inversion of a diagonal matrix is straightforward [23]. The Eigenvalue decomposition of positive definite matrices can also be done efficiently [23]. The pseudo-code of the algorithm is presented in Algorithm 1.

III. Experiments

We conducted experiments on robots with different architectures to emphasize two features of our method. First, the information loss induced by dimensionality reduction is not significant. Second, the expressive power of the linear

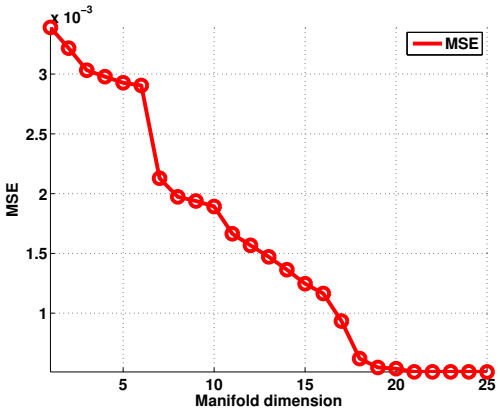


Fig. 5: The error of the direct correspondence transformation, measured in mean square error (MSE), applied for inverse dynamics learning. The dimension of the source dataset is $\dim(\mathbf{D}^s) = 32$ (joint angle, joint velocity, joint acceleration, torque value for every eight joint) and the dimension of the target dataset is $\dim(\mathbf{D}^t) = 28$ (the same values for seven joint). The error decreases as the dimension of the latent manifold increases but after a threshold (above 21) only the noise process is modeled.

function used for manifold alignment is sufficiently good to achieve good performance. The experiments were for tracking control learning and inverse dynamics learning.

A. Improve Inverse Dynamics with Direct Correspondence

The aim of this experiment is not to enhance any learning process in the target task, but to show that knowledge transfer is conceptually sound and can be done with good accuracy.

The experiment was performed on a simulated Sarcos Master arm [2] with eight DoFs and a simulated Barrett WAM arm [2] with seven DoFs – see Figure 1. The source task was task-space tracking of a trefoil knot (Figure 4c) while we wanted to see if the inverse dynamics learning of the Barrett arm can be improved. The target task was also to track a trefoil knot. The source and the target figures had different scales and were placed in different regions of the Euclidean space. We used the analytical controllers of the robots to collect data for both tasks. The data collection process can be replaced by any other process, e.g., a task shown by a human in the case of imitation learning.

The dimension of the source dataset was $\dim(\mathbf{D}^s) = 32$ (joint angle, joint velocity, joint acceleration, torque value for every eight joint) and the dimension of the target dataset was $\dim(\mathbf{D}^t) = 28$ (the same values for seven joint). We set the low dimensional representation to 18, i.e., $\dim(\mathbf{M}^s) = \dim(\mathbf{M}^t) = 18$. After dimensionality reduction, almost all of the variance[1] has been preserved in the low dimensional manifold for each task since the last couple of eigenvalues of the data were very close to zero.

After running each task with the same speed for one minute, we had data points with direct correspondence. The samples were collected at 480Hz with the Master arm and at 500Hz with the Barrett arm. Cubic spline interpolation was used both in the joint-space and torque-space to obtain direct point correspondence. We applied the method from Section II-B.1 on this dataset. Figure 3a and Figure 3a shows that after estimating $\mathbf{A}$ with Equation (3), we could transfer joint positions from the source dataset (red) to the target dataset (blue) with good efficiency (dashed green). For torque values the transformations are shown on Figures 4a-4b.

To see how much information can be caught if we do not assume direct correspondence, but only distribution minimization, we applied the method from Section II-B.2 to this dataset. These result are presented in Figure 3c and Figure 3d. It can be seen that the transformation of the joint values is not accurate, but it captures the correct mean and variance of the transferred data, as expected.

An important question is how to chose the dimension of the latent manifold. The accuracy of the transformation, measured in the mean square error (MSE), is shown on Figure 5 as a function of dimension of the latent manifold[2]. It can be seen that after a given value (21 in our experiments), the accuracy does not change. This phenomena means that dimensions above 21 models the noise process in the data. This noise process can be a result, e.g., of inaccurate measurements. As a result, setting the dimension of the latent manifold to a smaller value than the maximum possible leads to noise reduction. We propose to set this dimension of the manifolds to the maximum value where the noise is not modeled.

B. Direct Figure Transfer with Direct Correspondence

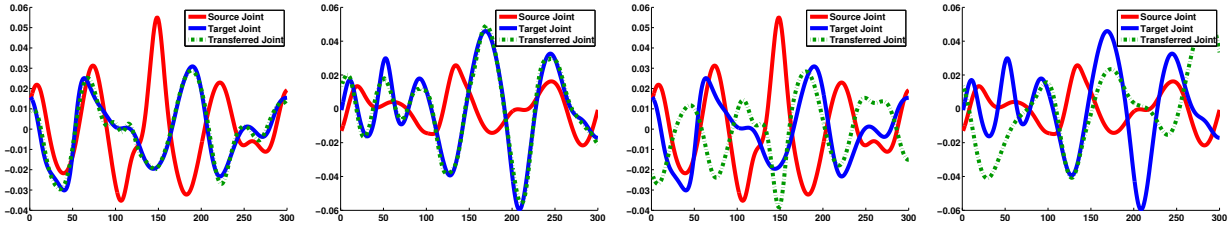
In this experiments, we used direct correspondence to match data for tracking control learning (kinematics data). Then, transferred a figure directly between two different architectures. To show that it is possible to transfer knowledge between robots with different DoFs, we enabled four joint of the Barrett arm and five joints of the Master arm[3]. Both source and target tasks were to do task-space tracking of a trefoil knot (Figure 4c). We again used the analytical controllers of the robots to collect data for both tasks. The source and the target figures had different scales and were placed in different regions of the Euclidean space. We set the dimension of the low dimensional representation to seven, i.e., $\dim(\mathbf{D}^s) = 8$, $\dim(\mathbf{D}^t) = 7$ (joint coordinates and end-effector coordinates), and $\dim(\mathbf{M}^s) = \dim(\mathbf{M}^t) = 7$.

We used a figure-eight test trajectory with the Master arm (using the analytical controller) that was placed inside the space defined by the trefoil knot. After transforming the joint-space trajectories and following the transformed trajectories with the Barrett arm, the figure eight presented in Figure 6a has been obtained. Note that showing a desired figure eight

[1]The loss of variance was below numerical precision of Matlab.

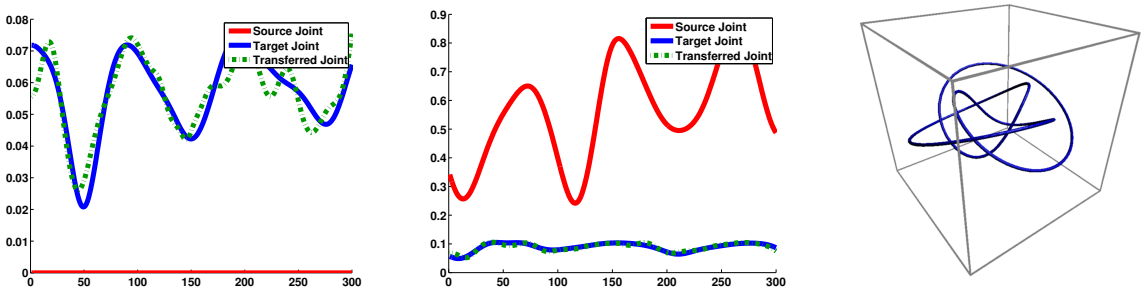
[2]The mean square error is relatively small (10^{-3}) since some joints have been used to perform the task, thus, their value was very close to zero. This property pulls the error closer to zero but does not affect the trend as a function of the dimensions.

[3]We experimented with more joints enabled as well, however, no improvement on the tracking accuracy has been observed.



(a) Joint 1. (b) Joint 2. (c) Joint 1. (d) Joint 2.

Fig. 3: Using direct correspondence (a) and (b), the method could find an accurate mapping (dashed green curves) from the source joints (red curves) to the target joints (blue curves) for all DoF. With rough alignment (c) and (d) the method could not find an accurate mapping from the source joints to the target joints. However, the mean and the variance for all the target joints are well estimated, as expected.



(a) Torque 5. (b) Torque 6. (c) Trefoil knot.

Fig. 4: Using direct correspondence (a) and (b) shows the transferred torque values (green dashed curves) from the Master arm (red curves) to the Barrett arm (blue curves) when applied for inverse dynamics. The transformation is accurate enough.

in Figure 6a may be misleading since there is no ground truth of task transformation. We defined the desired figure eight as the figure tracked by the analytical controller of the Barrett arm with the same initial posture as the trefoil knot.

C. Speed-up Tracking Control Learning using Different Robot Architectures

In this experiment, we used the same two robot architectures as in the previous experiment. We used the source dataset from the previous experiment (the trefoil knot). The target task was to speed-up the tracking control learning of the Barret arm. We set the dimension of the low dimensional representation again to seven, thus, $\dim(\mathbf{D}^{\mathrm{s}}) = 8$, $\dim(\mathbf{D}^{\mathrm{t}}) = 7$, and $\dim(\mathbf{M}^{\mathrm{s}}) = \dim(\mathbf{M}^{\mathrm{t}}) = 7$.

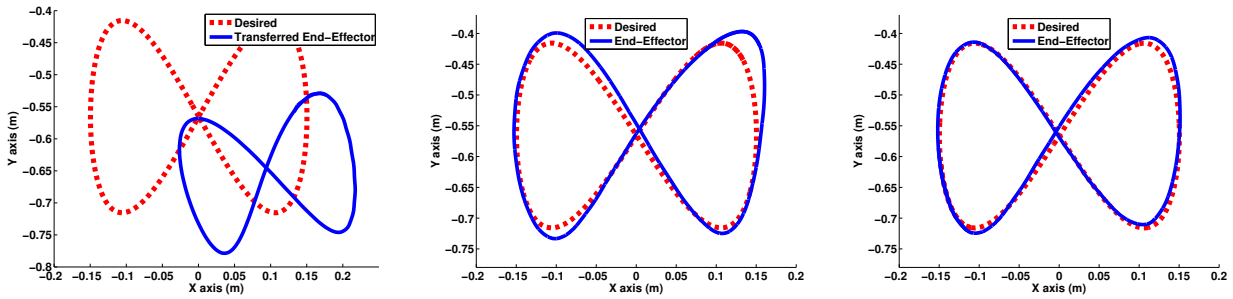
The tracking control algorithm was based on the the forward kinematics model of the robot. The forward kinematics model has been approximated with sparse online Gaussian processes and used to perform task-space control [25].

Without transfer learning, it takes from 20 seconds to four minutes to learn this model online [25]. After performing quasi-random movements for three seconds with the Barrett arm, we applied the distribution alignment approach presented in Section II-B.2 between the Master arm dataset and the collected one. We stopped the learning process after the three seconds of burn-in period and used the transferred points to further train the forward kinematics model. Figure 6b shows the figure eight as a result. The shape eight is not perfect, however, we needed only *three seconds* of learning and the transfer algorithm. We repeated the experiment but now the learning process was not stopped after three seconds, only the Master arm dataset has been used to gain additional training samples. Figure 6c shows that if learning is not stopped an accurate figure eight tracking is achievable after three seconds of learning and the transfer algorithm. The use of the Master arm dataset was necessary in the experiment. Based only on the data collected during the first three seconds of interaction, the arm failed to learn the forward kinematics model.

IV. Discussion

We conclude that transfer learning is a natural way of extending the limits of standard supervised, unsupervised, or reinforcement learning frameworks. Experiments show that once a mapping between two robot architectures is known, direct task transfer between these robots is possible. Furthermore, significant improvement on learning forward kinematics can be achieved when datasets from past experiments are also considered.

The presented method has limitations. When rough alignment is applied, assuming Gaussian distributed data may be too restrictive and more complex distribution should also be considered. This extension would come at the cost of increasing computational complexity or even losing analytical tractability. Another possible improvement may be to take into account the sequential nature of data and use this property in the distribution minimization process. Considering



(a) Direct task transfer from the Master arm to the Barrett arm.

(b) Figure eight after data transfer *without* further learning.

(c) Figure eight after data transfer *with* further learning.

Fig. 6: Tracking control results after three seconds of online learning, then, using the transfer learning algorithm: (a) when no further online model learning is allowed after data transfer; (b) when online model learning is allowed after data transfer. (c) The figure eight drawn by the Master arm has been directly transferred to the Barrett arm.

constrained distribution minimization would almost certainly lead to analytical intractability, thus, slower algorithm.

Experiments show that the class of linear functions has a good expressive power to model the relationship between the manifolds. Releasing this linearity assumption and considering more complex relationships may lead to over-fitting.

To see the limitations of the linear mapping, experiments should be conducted on more complex (humanoid) robots with other robot models, e.g., inverse kinematics or dynamics models, as well. It is important to see where the transformation breaks down, i.e., how different the robot architectures can still have a meaningful transformation. Transferring knowledge between different models, e.g., between kinematics and dynamics models, also appears to be a challenging task.

Acknowledgements

B. Bócsi acknowledges for the financial support provided by the POSDRU 88/1.5/S/60185. B. Bócsi and L. Csató acknowledge the support of the Romanian Ministry of Education, grant PN-II-RU-TE-2011-3-0278.

Part of the research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreements no. ICT-270327 (CompLACS) and no. ICT-600716 (CoDyCo).

References

[1] D. Nguyen-Tuong and J. Peters, "Model learning in robotics: a survey," *Cognitive Processing*, 10 2011.

[2] D. Nguyen-Tuong, M. W. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[3] H. D. Patino, R. Carelli, and B. R. Kuchen, "Neural networks for advanced control of robot manipulators." *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 343–354, 2002.

[4] B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters, "Learning inverse kinematics with structured prediction," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011, pp. 698–703.

[5] K. M. Chai, C. Williams, S. Klanke, and S. Vijayakumar, "Multi-task Gaussian Process Learning of Robot Inverse Dynamics," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 265–272.

[6] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*. Piscataway, NJ: IEEE, 2001.

[7] C. Salaun, V. Padois, and O. Sigaud, "Learning forward models for the operational space control of redundant robots." in *From Motor Learning to Interaction Learning in Robots*, O. Sigaud and J. Peters, Eds., vol. 264. Springer, 2010, pp. 169–192.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowledge and Data Engin.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[9] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer Learning via Dimensionality Reduction," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

[10] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. 1, pp. 1633–1685, 2009.

[11] S. B. Thrun and T. M. Mitchell, "Lifelong Robot Learning," Tech. Rep. IAI-TR-93-7, January 1993.

[12] W. Dai, Q. Yang, G. R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning (ICML)*. ACM, 2007, pp. 193–200.

[13] W. Dai, G. rong Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 2007, pp. 540–545.

[14] E. V. Bonilla, K. M. Chai, and C. K. I. Williams, "Multi-task Gaussian process prediction," in *Advances in Neural Information Processing Systems 20 (NIPS)*, J. C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008.

[15] R. Raina, A. Y. Ng, and D. Koller, "Constructing informative priors using transfer learning," in *In Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 713–720.

[16] J. Huang, A. Gretton, B. Schölkopf, A. J. Smola, and K. M. Borgwardt, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2007.

[17] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, vol. 16, pp. 307–354, 1992.

[18] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison," *Int. J. Rob. Res.*, vol. 27, no. 6, pp. 737–757, June 2008.

[19] D. Nguyen-Tuong and J. Peters, "Learning task-space tracking control with kernels," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 704–709.

[20] J. Peters and S. Schaal, "Learning operational space control," in *Robotics: Science and Systems (RSS 2006)*. MIT Press, 2006.

[21] J. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[22] C. Wang and S. Mahadevan, "Manifold alignment using Procrustes analysis," in *Proceedings of the 25th international conference on Machine learning*. ACM New York, NY, USA, 2008, pp. 1120–1127.

[23] L. Trefethen and D. Bau, *Numerical linear algebra*, ser. Miscellaneous Bks. Society for Industrial and Applied Mathematics, 1997.

[24] S. Kullback, *Information Theory and Statistics*. New York: Wiley, 1959.

[25] B. Bócsi, P. Hennig, L. Csató, and J. Peters, "Learning tracking control with forward models," in *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, USA, 2012.