2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Independent Joint Learning: A Novel Task-to-Task Transfer Learning Scheme for Robot Models

Terry Taewoong Um, Myoung Soo Park, and Jung-Min Park

Abstract— **In the past decade, model learning techniques have provided appealing approaches for determining the dynamic model of robots from data. These techniques strongly capture the complicated effects of robot dynamics, which are often neglected in hand-crafted dynamic models. However, unlike robust performance shown in trained tasks, learned models do not exhibit a reliable performance in new tasks as they are valid only near the domain of the trained tasks.**

In this paper, we propose an alternative approach for task-to-task transfer learning, called "Independent Joint Learning (IJL)." IJL learns the model for each joint independently rather than the whole body at one time to effectively transfer knowledge between tasks. A comparative simulation study on a 6 DOF PUMA robot demonstrates that our approach outperforms other related approaches when a task different from trained tasks is proposed.

I. Introduction

Model-based control strategies have shown many advantages over traditional PD control such as improved tracking accuracy, reduced energy consumption, and enabled compliant movements [1]. Model-based control often requires an inverse dynamic model, which defines the required forces and torques based on the robot's given kinematic and dynamic information.

$$\boldsymbol{f} : (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \rightarrow \boldsymbol{\tau} \tag{1}$$

One of the most frequently-used equations for determining the inverse dynamic model is the rigid body dynamics (RBD) equation:

$$\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{G}(\boldsymbol{q}) \tag{2}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^{\boldsymbol{n}}$ denotes joint positions, velocities, and accelerations, respectively, and $\boldsymbol{\tau}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{\boldsymbol{n}}$ indicates the corresponding torque values [2].

Acquiring an accurate robot model is a very crucial step for the robust performance of model-based robot control. However, it can become a challenging problem when the robot includes non-rigid components (e.g. hydraulic tube, elastic cable, or soft components) that cannot be modeled by using (2). Furthermore, the complexity of frictions and actuator dynamics may cause handcrafted model errors that degrade the general robot task performance.

Terry Taewoong Um is with Korea Institute of Science and Technology (KIST), Seoul, Republic of Korea. (e-mail: terry.t.um@gmail.com)

Myoung Soo Park is with Korea Institute of Science and Technology (KIST), Seoul, Republic of Korea. (corresponding author, phone: +82-2-958-6813; fax: +82-2-958-6813; e-mail: meister1@gmail.com).

Jung-Min Park is with Korea Institute of Science and Technology (KIST), Seoul, Republic of Korea. (e-mail: pjm@kist.re.kr)

In this respect, nonparametric model learning techniques like Gaussian Process Regression (GPR) [3] and Locally Weighted Projection Regression (LWPR) [4] are appealing alternatives to determine a robot model from data [5]. Instead of using a parametric closed-form solution like (2), these model learning methods attempt to directly learn the relationship between the robot states and the required torques in (1). The methods show the complex robot model capture and successful applications to various model-based control schemes such as inverse dynamic control, feed forward control [6], and operational space control [7].

To build a model for a specific task, gathering and training a sufficient amount of data near the domain of the task is required. However, this process is not only time-consuming but also task-specific. Since the learned model is valid only near the domain of the trained tasks, a new dissimilar task requires another gathering and training process. If we are able to reuse the knowledge from the already learned tasks, we will reduce the unnecessary gathering and training for the new task. This concept of transferring knowledge between different tasks or domains is called transfer learning [8].

Since robots are expected to complete not only trained tasks but also extensive tasks, there is a high demand for knowledge transfer between different tasks. Yet in spite of its necessity, there has been few research conducted on task-to-task transfer learning for robot models. Although Bócsi et al. [9] recently proposed a transfer learning scheme for robot models, it was about robot-to-robot transfer learning rather than task-to-task transfer learning. Thus, a transfer learning scheme between different tasks will provide substantial benefits to a wide range of robots.

In this paper, we propose a novel task-to-task transfer learning scheme for robot models, called "Independent Joint Learning (IJL)." By taking the benefits of the recursive Newton-Euler formulation, we develop an alternative learning approach for robot models that improves the knowledge transfer for a new task.

The paper will be organized as follows: We will first briefly overview the model learning approaches with a standard GPR and give a general introduction to transfer learning for robot models. In Section III, we explain the limited transferring ability of the existing model learning methods and suggest a novel transfer learning scheme. In Section IV, the new method is evaluated on a simulated 6 DOF manipulator and compared to existing model learning methods. Finally, the conclusion and future works are discussed in Section V.

II. Background

A. Model Learning with a standard GPR

The inverse dynamic model for model-based control is described as the mapping from joint positions, velocities, and accelerations to torques, as shown in (1). The aim of model learning is to predict the torque value of the i^{th} joint, $\tau_i \in \mathbb{R}$, as the response of the query point at $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{3m\times 1}$ by using the given n training data, $\{\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j], \tau_i[j]\}_{j=1}^{n}$, where $\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j] \in \mathbb{R}^{m\times 1}$ and $\tau_i[j] \in \mathbb{R}$. Since the problem can be considered as a supervised learning problem, any supervised learning technique can be employed for the learning process.

Gaussian process regression (GPR), which is a global supervised learning method, is one of the most widely-used techniques for model learning problems. The goal of GPR is to find the function f that maps the input vectors $\{\boldsymbol{x}[j]\}_{j=1}^{n}$ to the output values $\{y[j]\}_{j=1}^{n}$ where $\boldsymbol{x}[j] \in \mathbb{R}^{\boldsymbol{m}\times 1}$ and $y[j] \in \mathbb{R}$. In other words, the objective of GPR is to determine $y = f(\boldsymbol{x}) + \varepsilon$ for the query point $(\boldsymbol{x}, y)$ where $\boldsymbol{x} \in \mathbb{R}^{m\times 1}, y \in \mathbb{R}$, and $\varepsilon \in \mathbb{R}$ is Gaussian noise with zero mean and variance σ_n^2.

Given a set of n training data $\{\boldsymbol{x}[j], y[j]\}_{j=1}^{n}$, the observed targets can be described by a Gaussian distribution [3].

$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I}) \quad (3)$$

where $\boldsymbol{X} = [\boldsymbol{x}[1]\ \boldsymbol{x}[2] \cdots \boldsymbol{x}[n]] \in \mathbb{R}^{m\times n}$,

$$\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) = \begin{bmatrix} k(\boldsymbol{x}[1],\boldsymbol{x}[1]) & \cdots & k(\boldsymbol{x}[1],\boldsymbol{x}[n]) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}[n],\boldsymbol{x}[1]) & \cdots & k(\boldsymbol{x}[n],\boldsymbol{x}[n]) \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$\mathbf{K}(\boldsymbol{X},\boldsymbol{X})$ is the covariance matrix evaluated by a covariance function, $k(\boldsymbol{x}[p], \boldsymbol{x}[q])$, for all pairs of the input $p, q \in \{1,2,\cdots,n\}$. One of the commonly-used covariance functions is the squared exponential (SE) function given as

$$k(\boldsymbol{x}[p], \boldsymbol{x}[q]) = \theta_1 \exp(-\theta_2 ||\boldsymbol{x}[p] - \boldsymbol{x}[q]||^2) \quad (4)$$

where $\theta_1, \theta_2 \in \mathbb{R}$ are the hyperparameters that modulate the amplitude and characteristic length-scale, respectively.

With the query point $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{m}\times 1}$, the joint distribution of the observed targets $\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{n}\times 1}$ and predicted output $f(\boldsymbol{x}) \in \mathbb{R}$ is

$$\begin{bmatrix} \boldsymbol{y} \\ f(\boldsymbol{x}) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} + \sigma_n^2\boldsymbol{I} & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & k \end{bmatrix}\right) \quad (5)$$

where $\mathbf{K} = \mathbf{K}(\boldsymbol{X},\boldsymbol{X}) \in \mathbb{R}^{n\times n}$, $\mathbf{k} = \mathbf{k}(\boldsymbol{X},\boldsymbol{x}) \in \mathbb{R}^{n\times 1}$, and $k = k(\boldsymbol{x},\boldsymbol{x}) \in \mathbb{R}$. The predicted mean $\bar{f}(\boldsymbol{x})$ and variance $\bar{V}(\boldsymbol{x})$ is obtained from the aforementioned distribution [3].

$$\bar{f}(\boldsymbol{x}) = \mathbf{k}^{\mathrm{T}}(\mathbf{K} + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{y} \quad (6)$$

$$\bar{V}(\boldsymbol{x}) = k - \mathbf{k}^{\mathrm{T}}(\mathbf{K} + \sigma_n^2\boldsymbol{I})^{-1}\mathbf{k} \quad (7)$$

For model learning problems, the predicted mean $\bar{f}(\boldsymbol{x})$ is often taken as the required torque at the given state $\boldsymbol{x}$. Given n training data, $\{\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j], \tau_i[j]\}_{j=1}^{n}$, the required torque value of the i^{th} joint, $\bar{f}(\boldsymbol{x})$, is obtained from (6), where $\boldsymbol{x} = [\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}] \in \mathbb{R}^{3m\times 1}$, $\boldsymbol{y} = [\tau_i[1], \cdots, \tau_i[n]] \in \mathbb{R}^{n\times 1}$ and

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{q}[1] & \cdots & \boldsymbol{q}[n] \\ \dot{\boldsymbol{q}}[1] & \cdots & \dot{\boldsymbol{q}}[n] \\ \ddot{\boldsymbol{q}}[1] & \cdots & \ddot{\boldsymbol{q}}[n] \end{bmatrix} \in \mathbb{R}^{3m\times n} \quad (8)$$

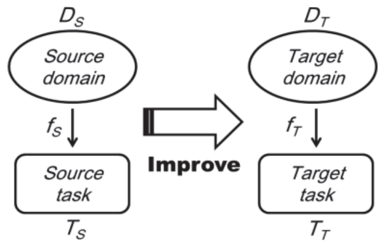

Figure 1. A breif scheme of task-to-task transfer learning. The transfer learning aims to help the target learning process by using the knowledge from the source task in the source domain.

The learned model obtained by using GPR shows a robust performance in predicting torques when the query point $\boldsymbol{x}$ is not far from the training input $\boldsymbol{X}$. However, when it is far, the learned model often fails to provide accurate predictions and additional training processes is required to improve the accuracy. If we are able to reuse the relevant knowledge from past tasks, we would not have to go through unnecessary efforts. This is the reason why we try to introduce the concept of transfer learning for model learning problems.

B. Transfer Learning for Robot Models

The main purpose of transfer learning is to improve the target learning process by transferring the knowledge of previously learned tasks. Let D_S, T_S, D_T, and T_T be a source domain, source task, target domain, and target task, respectively. The objective of transfer learning is to improve the learning of the target prediction function f_T for T_T in D_T by using the knowledge provided in D_S and T_S. (See *Figure 1* and [8])

The transfer learning research examples can be found in web-document classification [10], WiFi localization [11], and reinforcement learning [12] among others. For transfer learning in robot models, Bócsi et al. [9] attempted to transfer knowledge between different robots that have different robot architecture. They hoped to enhance the tracking task performance of a 7 DOF Barrett WAM by using the knowledge gained from another robot, 8 DOF Sarcos Master, which performed the same tracking task previously.

Unlike the previous work, our approach aims at task-to-task transfer learning problems where the source and target tasks are different while operating robots are identical. This is different from multi-task learning in that multi-task learning seeks to improve the performance of all tasks simultaneously while transfer learning is more focused on the target task. To the best of our knowledge, our approach is among the first attempts to deal with task-to-task transfer learning for robot models. Since robots are often expected to fulfill various tasks, task-to-task transfer learning provides substantial advantages to real world applications.

In this paper, a novel task-to-task transfer learning scheme for robot models is suggested under the assumption that imperfect prior knowledge on the robot model exists. In particular, we postulate that significant errors are included in the measured inertial parameters. This assumption sounds reasonable because inertial parameters are more difficult to measure than kinematic ones. In the next section, we will see the motivation behind our idea and detailed specifications of the novel approach, IJL.

III. INDEPENDENT JOINT LEARNING FOR A NEW TASK

A. Motivation

We can learn from human behavior when coping with a new task in that humans reuse only a portion of previously learned knowledge for a new task rather than exploiting entirely with the irrelevant parts. For example, if basketball players trained for the high jump to enhance their jumping skills, they may reuse the now-learned ankle and knee movements for the jumping motion in a basketball play rather than reuse other trained movements (e.g., wrist or elbow movements). Humans consider the trained knowledge of each joint motion separately and selectively reuse them for a new task.

Thus, model learning approaches also need to train the data of each joint separately and selectively reuse them for a new task. To achieve this, the training and target data need to expose their relevancy so that we can discern which training data is reusable for the target task by inspecting the data.

The desired properties for the training and target data in an effective transfer learning are as follows:

- The training and target data for joint *i* appear to be close to each other if the training motions of joint *i* are closely related to the target motions of joint *i*.
- The training and target data for joint *i* appear to be far from each other if the training motions of joint *i* are unrelated to the target motions of joint *i*.
- The knowledge of the selected joints can be reused for the target task, independently, based on their relevancy. For example, the ankle joint knowledge is reused for the high jump while the wrist joint knowledge is not.
- The irrelevant joint motions do not interfere with the learning process of the relevant joint motions. For example, the wrist joint motions do not interfere with the learning process of the ankle joint in the high jump training.

However, traditional model learning approaches that learn the mapping from positions, velocities, and accelerations of all joints to each joint torque, $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \rightarrow \tau_i$, have the far aspects from the desired properties presented above. (For convenience, we will call this approach as the "Lagrangian-based approach" in the rest of the paper).

First, the relevant and irrelevant joint knowledge is not distinguishable in the Lagrangian-based approach, because all joints have the same training input $\boldsymbol{X}$. If the training data of the i th joint are reusable but the others are not, then the training data of the i th joint are close to the target data of the i th joint so that they betray their close relevancy. In addition, the training data of the other joints may be expected to be far from the target data of the corresponding joints because they are irrelevant to the target task.

However, this situation does not happen in the Lagrangian -based approach, because all training input $\boldsymbol{X}$ are the same regardless of the joint. Thus, there exist only two cases in the Lagrangian-based approaches: when the $\boldsymbol{X}^{train}$ is close to $\boldsymbol{X}^{target}$ so that the training data for all joints are reusable, or when $\boldsymbol{X}^{train}$ is far from $\boldsymbol{X}^{target}$ so that the training data for all joints cannot be reused. The case when only specific joint knowledge is reusable cannot be handled in the Lagrangian-based approach, since the relevant and irrelevant joints are indistinguishable under the representation of the approach.

Moreover, in the Lagrangian-based approach, the irrelevant joint motions may interfere with the learning process of the motions of the relevant joint. When a learned model is built simply based on the position data of the ankle, knee, and wrist joints, then

$$\boldsymbol{X} = \{\boldsymbol{q}_{ankle}[j],\ \boldsymbol{q}_{knee}[j],\ \boldsymbol{q}_{wrist}[j]\}_{j=1}^{n} \tag{9}$$

If we use the training data for jumping motion generation, the ankle and knee joint elements will provide more important knowledge than the wrist joint elements. Since the learned model reflects each element of the training data equivalently, the wrist elements in the training data can disturb the learning process of the ankle and knee motions. For example, if the training and target data have the same ankle and knee values but different wrist values, they may turn out to be dissimilar. However, they have to be treated as if they are close in the jumping motion training.

The learned model obtained from the Lagrangian-based approach shows a lack of transferability for a new task. One of the principal reasons for the lack of transferability is that the Lagrangian-based approach learns the entire body dynamics at once rather than each joint dynamics separately. As the learned model contains superfluous components like redundant information, it is likely to exhibit a poor performance in a new task.

Thus, for an extensive transferability of the learned model, it needs to be broken down into smaller parts and selectively reused for a new task. In the next section, we will describe the process of breaking down the dynamic model into smaller parts by using the Newton-Euler formulation. In addition, we will introduce the novel learning approach, the IJL, for an effective model knowledge transfer between tasks.

B. Independent joint learning for robot dynamic model

The rigid body dynamic model is defined by the equations of motion derived from the Lagrangian or Newton-Euler (N-E) formulations. While the Lagrange formulation determines the dynamic model by using the total Lagrangian of the robot system, the N-E formulation sequentially determines each joint model based only on the forces and torques acting on it [2]. In other words, the N-E formulation is appropriate for creating modular models, which is beneficial for robot model transfer learning.

In N-E formulation, a forward recursion is performed for propagating velocities and accelerations. A backward recursion occurs from the last link as propagating forces and torques (See *Figure 2*)

The angular velocity and acceleration vector of (i+1)th link is calculated from the values of i th link as follows:

$${}^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}\boldsymbol{R}_i({}^{i}\boldsymbol{\omega}_i + \dot{q}_{i+1}\boldsymbol{z}_0) \tag{10}$$

$${}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}^{i+1}\boldsymbol{R}_i({}^{i}\dot{\boldsymbol{\omega}}_i + \ddot{q}_{i+1}\boldsymbol{z}_0 + {}^{i}\boldsymbol{\omega}_i \times (\dot{q}_{i+1}\boldsymbol{z}_0) \tag{11}$$

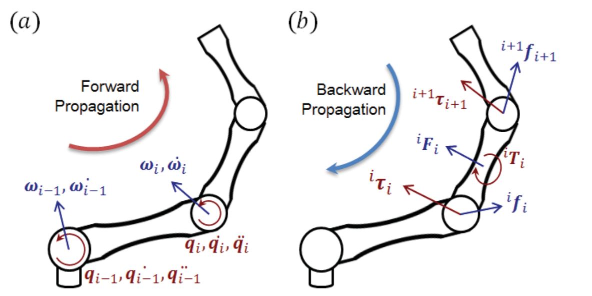

Figure 2. Forward(a) and backward(b) propagation in the recursive N-E algorithm. The torque of joint i is determined by the forces acting on i th link and the propagated forces from the distal links.

where $\boldsymbol{z}_0 = [0\ 0\ 1]^{\mathrm{T}}$ and ${}^{i+1}\boldsymbol{R}_i$ represent the 3×3 rotation matrix defining frame i with respect to frame $i+1$. The upper-left and lower-right superscript of the letter denotes the reference frame and joint number, respectively.

The force ${}^{i}\boldsymbol{F}_i \in \mathbb{R}^{3\times1}$ and moment ${}^{i}\boldsymbol{T}_i \in \mathbb{R}^{3\times1}$ at the center of mass (COM) of the i th link are

$$ {}^{i}\boldsymbol{F}_i = m_i {}^{i}\dot{\boldsymbol{v}}_{COM_i} \tag{12} $$

$$ {}^{i}\boldsymbol{T}_i = \boldsymbol{I}_i {}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i}\boldsymbol{\omega}_i \times (\boldsymbol{I}_i {}^{i}\boldsymbol{\omega}_i) \tag{13} $$

where $\boldsymbol{I}_i$ represents the 3×3 inertia matrix of the i th link and ${}^{i}\dot{\boldsymbol{v}}_{COM_i} \in \mathbb{R}^{3\times1}$ represents the acceleration at the COM of the i th link, which is a function of ${}^{i}\boldsymbol{\omega}_i$ and ${}^{i}\dot{\boldsymbol{\omega}}_i$ [2].

Finally, the force ${}^{i}\boldsymbol{f}_i \in \mathbb{R}^{3\times1}$ and torques ${}^{i}\boldsymbol{\tau}_i \in \mathbb{R}^{3\times1}$, $\tau_i \in \mathbb{R}$ acting on the i th joint can be evaluated by

$$ {}^{i}\boldsymbol{f}_i = {}^{i}\boldsymbol{R}_{i+1} {}^{i+1}\boldsymbol{f}_{i+1} + {}^{i}\boldsymbol{F}_i \tag{14} $$

$$ \begin{aligned} {}^{i}\boldsymbol{\tau}_i = &({}^{i}\boldsymbol{r}_i + {}^{i}\boldsymbol{r}_{COM_i}) \times {}^{i}\boldsymbol{F}_i + {}^{i}\boldsymbol{T}_i + \\ &+ {}^{i}\boldsymbol{R}_{i+1}\{{}^{i+1}\boldsymbol{\tau}_{i+1} + ({}^{i+1}\boldsymbol{R}_i\ {}^{i}\boldsymbol{r}_i) \times {}^{i+1}\boldsymbol{f}_{i+1}\} \end{aligned} \tag{15} $$

$$ \tau_i = {}^{i}\boldsymbol{\tau}_i \cdot ({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0) + \varepsilon \tag{16} $$

where ${}^{i}\boldsymbol{r}_i \in \mathbb{R}^{3\times1}$ is the displacement from $(i-1)$th joint to i th joint, ${}^{i}\boldsymbol{r}_{COM_i} \in \mathbb{R}^{3\times1}$ is the position vector of the COM of i th link, and $\varepsilon \in \mathbb{R}^{3\times1}$ represents the error between the actual and calculated values. ${}^{i+1}\boldsymbol{\tau}_{i+1}$ and ${}^{i+1}\boldsymbol{f}_{i+1}$ are propagated from the previous link. (These values are zero for the last link).

In summary, the torque vector ${}^{i}\boldsymbol{\tau}_i$ consists of two parts: torques caused by the current link's movement and torques propagated from the distal links.

Current link's torques :

$$ ({}^{i}\boldsymbol{r}_i + {}^{i}\boldsymbol{r}_{COM_i}) \times {}^{i}\boldsymbol{F}_i + {}^{i}\boldsymbol{T}_i = f(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i) \tag{17} $$

Propagated torques :

$$ {}^{i+1}\boldsymbol{\tau}_{i+1} + ({}^{i}\boldsymbol{R}_{i+1} {}^{i}\boldsymbol{r}_i) \times {}^{i+1}\boldsymbol{f}_{i+1} = f(\boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}) \tag{18} $$

Sum of the torques acting on the joint i :

$$ {}^{i}\boldsymbol{\tau}_i = \text{Current link's torques} + \text{Propagated torques} $$

As mentioned in the introduction, non-rigid components or inaccurate robot parameters in the robot model can mean the differences between the analytic RBD model and real model. However, the RBD model takes a major role in governing robot movements. Additionally, it is natural to expect that the RBD model error to relate to the current RBD model state. In this respect, the error term can be considered as a function of $\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}$, and $\boldsymbol{f}_{i+1}$. Thus, the real torque value of the joint i can be predicted as

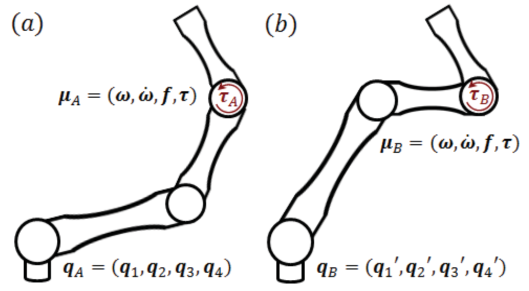
(a)
(b)
$\mu_A = (\boldsymbol{\omega}, \dot{\boldsymbol{\omega}}, \boldsymbol{f}, \boldsymbol{\tau})$
τ_A
$q_A = (q_1, q_2, q_3, q_4)$
τ_B
$\mu_B = (\boldsymbol{\omega}, \dot{\boldsymbol{\omega}}, \boldsymbol{f}, \boldsymbol{\tau})$
$q_B = (q_1', q_2', q_3', q_4')$

Figure 3. Let us suppose a case where all joints stay at rest except the last joint. Since the conditions for the last joints in (a) and (b) are identical, $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_B$ will exhibit the same values. However, the Lagrangian-based approach fails to predict the torques because $\boldsymbol{q}_A$ and $\boldsymbol{q}_B$ are dissimilar whereas the IJL provides a correct prediction.

$$ \tau_i = {}^{i}\boldsymbol{\tau}_i \cdot ({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0) + \varepsilon(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1}). \tag{19} $$

In the IJL approach, the mapping from $(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1})$ to $\varepsilon_i = \tau_i^{true} - \tau_i$ is trained by a standard GPR. For a new task, the IJL method provides the target torque prediction based on the RBD model and GPR estimated error. (Details of the method are provided in *Algorithms 1, 2,* and *3*)

The IJL approach learns the errors in terms of $\varepsilon(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1})$ instead of $\varepsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, as in [13]. Thus, we now have different training sets for each joint able to transfer the knowledge from joint to joint. In the next section, the IJL approach is evaluated through simulations.

Algorithm 1 Calculate Torques $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$

```
Input: new data point $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$
Initialize: ${}^{1}\boldsymbol{\omega}_1 \leftarrow \dot{q}_1\boldsymbol{z}_0$, ${}^{1}\dot{\boldsymbol{\omega}}_1 \leftarrow \ddot{q}_1\boldsymbol{z}_0$,
            ${}^{N}\boldsymbol{R}_{N+1} \leftarrow \boldsymbol{1}$, ${}^{N+1}\boldsymbol{\tau}_{N+1} \leftarrow \boldsymbol{0}$, ${}^{N+1}\boldsymbol{f}_{N+1} \leftarrow \boldsymbol{0}$
for i = 1 to N          (Forward recursion)
    ${}^{i+1}\boldsymbol{\omega}_{i+1} \leftarrow {}^{i+1}\boldsymbol{R}_i({}^{i}\boldsymbol{\omega}_i + \dot{q}_{i+1}\boldsymbol{z}_0)$
    ${}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \leftarrow {}^{i+1}\boldsymbol{R}_i({}^{i}\dot{\boldsymbol{\omega}}_i + \ddot{q}_{i+1}\boldsymbol{z}_0 + {}^{i}\boldsymbol{\omega}_i \times (\dot{q}_{i+1}\boldsymbol{z}_0)$
    ${}^{i}\boldsymbol{F}_i \leftarrow m_i {}^{i}\dot{\boldsymbol{v}}_i$ , ${}^{i}\boldsymbol{T}_i \leftarrow \boldsymbol{I}_i {}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i}\boldsymbol{\omega}_i \times (\boldsymbol{I}_i {}^{i}\boldsymbol{\omega}_i)$
end for
for i = N to 1          (Backward recursion)
    ${}^{i}\boldsymbol{\tau}_i \leftarrow ({}^{i}\boldsymbol{r}_i + {}^{i}\boldsymbol{r}_{CM}) \times {}^{i}\boldsymbol{F}_i + {}^{i}\boldsymbol{T}_i +$
        $+ {}^{i}\boldsymbol{R}_{i+1}\{{}^{i+1}\boldsymbol{\tau}_{i+1} + ({}^{i+1}\boldsymbol{R}_i {}^{i}\boldsymbol{r}_i) \times {}^{i+1}\boldsymbol{f}_{i+1}\}$
    $\tau_i \leftarrow {}^{i}\boldsymbol{\tau}_i \cdot ({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0)$
    $\mu_i \leftarrow (\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}, \tau_i)$
end for
Output: $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N)$
```

Algorithm 2 Independent Joint Learning : Training

```
Input: $\{\{\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j], \tau_i[j]\}_{j=1}^{n}\}_{i=1}^{N}$ , $\{\{\tau_i^{true}[j]\}_{j=1}^{n}\}_{i=1}^{N}$
for j = 1 to n
    $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N) \leftarrow$ CalculateTorques $(\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j])$
    for i = 1 to N
        $(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}, \tau_i) \leftarrow \mu_i$
        $\varepsilon_i[j] \leftarrow \tau_i^{true}[j] - \tau_i[j]$
        Training: $GPR^{Training}$ $\{(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}, \tau_i) \rightarrow \varepsilon_i[j]\}$
    end for
end for
```

Algorithm 3 Independent Joint Learning: Prediction

```
Input: $\{\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j]\}_{j=1}^{m}$
for j = 1 to m
    $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N) \leftarrow$ CalculateTorques $(\boldsymbol{q}[j], \dot{\boldsymbol{q}}[j], \ddot{\boldsymbol{q}}[j])$
    for i = 1 to N
        $(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}, \tau_i) \leftarrow \mu_i$
        Test: $\varepsilon_i[j] \leftarrow GPR^{Test}$ $\{(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{f}_{i+1}, \boldsymbol{\tau}_{i+1}, \tau_i)\}$
        $\tau_i^*[j] \leftarrow \tau_i + \varepsilon_i[j]$
    end for
end for
Output: $\{\boldsymbol{\tau}^*\}_{j=1}^{m} = \{(\tau_1^*[j], \tau_2^*[j], \cdots, \tau_N^*[j])\}_{j=1}^{m}$
```

IV. EXPERIMENTS

In this section, we compare the tracking performances by using the IJL and other model learning methods. The experiments are performed in simulation on a 6 DOF PUMA560 robot using the Matlab Robotics Toolbox [14].

A. Experiment Setting

For the simulation, we assume that the given RBD model includes +20% errors in mass and inertia matrix. For simplicity, the friction and motor actuation effects are neglected. The gravity vector direction is known so the RBD model uses it to calculate the gravity forces.

The open-source code of a standard GPR [3] is employed as a basic regressor to learn the inverse dynamic models, $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \rightarrow \boldsymbol{\tau}$ or $(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1}) \rightarrow \boldsymbol{\tau}$. The hyper-parameter values in (4) are determined by maximizing the log marginal likelihood for a given training dataset at each time

When evaluating the tracking performance of each method, we use sinusoidal trajectories with different frequencies and amplitudes for each joint, which are also used in [6].

$$q_i(t) = A_i sin(\omega_{1_i} t) + A_i/3 sin(\omega_{2_i} t) \qquad (20)$$

To check the effect of the training data numbers, a single task is used for training in the first experiment whereas three tasks are used in the second (See *Appendix*).

TABLE I. APPROXIMATION ERROR AS MSE FOR EACH JOINT TORQUE (TRAINED TASK : A / TARGET TASK : D)

Joint Torque [Nm]	GPR only w/ $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$	RBD only	RBD+GPR w/ $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$	RBD+GPR w/ $(\boldsymbol{\omega}, \dot{\boldsymbol{\omega}}, \boldsymbol{\tau}, \boldsymbol{f})$
$\boldsymbol{\tau}_1$	21.854	0.6232	0.5926	0.1399
$\boldsymbol{\tau}_2$	154.322	46.498	4.5408	0.7928
$\boldsymbol{\tau}_3$	60.601	1.223	0.5539	0.3730
$\boldsymbol{\tau}_4$	4.425×10^{-5}	1.180×10^{-6}	1.316×10^{-6}	1.061×10^{-5}
$\boldsymbol{\tau}_5$	6.235×10^{-4}	1.737×10^{-5}	1.726×10^{-5}	2.549×10^{-5}
$\boldsymbol{\tau}_6$	1.046×10^{-4}	2.245×10^{-10}	1.101×10^{-4}	5.402×10^{-20}
$\boldsymbol{\tau}_{mean}$	39.463	8.057	0.9479	0.2176

TABLE II. APPROXIMATION ERROR AS MSE FOR EACH JOINT TORQUE (TRAINED TASK : A,B,C / TARGET TASK : D)

Joint Torque [Nm]	GPR only w/ $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$	RBD only	RBD+GPR w/ $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$	RBD+GPR w/ $(\boldsymbol{\omega}, \dot{\boldsymbol{\omega}}, \boldsymbol{\tau}, \boldsymbol{f})$
$\boldsymbol{\tau}_1$	161.948	0.6232	0.3664	0.0893
$\boldsymbol{\tau}_2$	54.812	46.498	16.897	0.4076
$\boldsymbol{\tau}_3$	1.449	1.223	0.0381	0.1068
$\boldsymbol{\tau}_4$	4.257×10^{-4}	1.180×10^{-6}	8.503×10^{-6}	7.054×10^{-7}
$\boldsymbol{\tau}_5$	5.360×10^{-4}	1.737×10^{-5}	3.240×10^{-5}	1.979×10^{-5}
$\boldsymbol{\tau}_6$	4.563×10^{-4}	2.245×10^{-10}	5.193×10^{-7}	5.727×10^{-24}
$\boldsymbol{\tau}_{mean}$	36.368	8.057	2.8836	0.1006

The IJL approach is compared to three other methods. One is a common model learning approach that directly learns the inverse dynamic model $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \rightarrow \boldsymbol{\tau}$ without any a-priori knowledge. There is also an imperfect RBD model method and a-priori knowledge RBD method that learns the error parts in the $\varepsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ form, as presented in [13].

B. IJL approach Evaluation

We compare the tracking performances of the traditional model learning method, which learns the inverse dynamics $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \rightarrow \boldsymbol{\tau}$, and the IJL approach, which learns only the errors in the form of $\varepsilon(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1}) \rightarrow \boldsymbol{\tau}$. As we can see in *Table I* and *Table II*, the traditional method is poor in comparison with the IJL approach. This result coincides with our expectation that the traditional method will be poor when the training and target data are dissimilar.

We also compared the tracking performances between the $\varepsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ and $\varepsilon(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1})$ parameterizations when an imperfect RBD model is given. Among the three cases, "RBD only" has the worst performance since it does not have an error correction methodology. The IJL approach outperforms other methods because it learns each joint dynamics separately and transfers it from joint to joint.

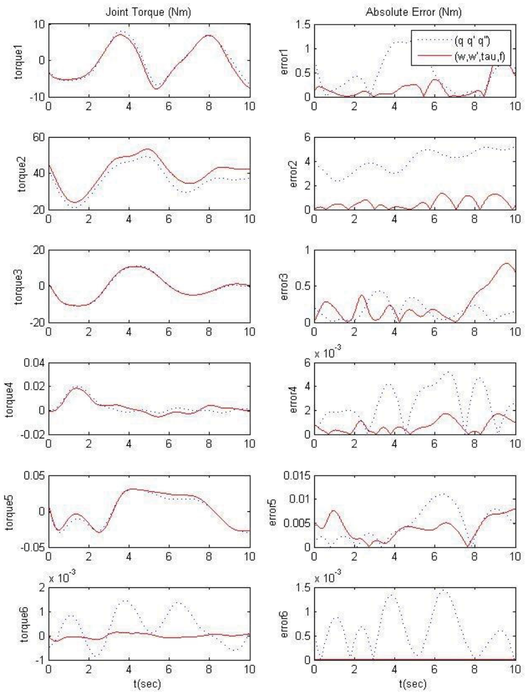

Figure 4. Tracking performance of $\varepsilon(\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{\tau}_{i+1}, \boldsymbol{f}_{i+1})$ and $\varepsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ parameterizations when Task A,B,C and Task D are given as the training and the task set, respectively.

Another advantage of the IJL is that it shows a noticeably better performance when the number of training tasks is increased. This means that the IJL approach transfers the knowledge from the training tasks to a new task more effectively than other approaches. Unlike the Lagrangian -based approach where the training and target domain's dimensions increase as 3N (N DOF for each $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$), the IJL approach's dimensions remain twelve (3 DOF for each $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{\omega}}_i$, $\boldsymbol{\tau}_{i+1}$, and $\boldsymbol{f}_{i+1}$) regardless of the robot DOF. For the robots that have more than 4 DOF, the IJL approach provides a better chance for overlapping data and is likely to show better performance in a new task.

V. Conclusion

In this paper, we propose a novel model learning approach, called "Independent Joint Learning (IJL)." By learning the dynamic model for each joint independently, the IJL is better in task-to-task transfer learning and in a new task. Additionally, comparative experiments with different model learning approaches are performed in simulation.

Based on this research, future research will focus on the methodology of the knowledge transfer between different tasks. By illuminating the training data that is transferrable or nontransferable for the target task, future research will prevent negative transfer [15] and maximize the effectiveness of transfer learning.

Appendix

TABLE III. Values taken for training and test trajectories $q_i(t) = A_i sin(\omega_{1_i} t) + A_i/3sin(\omega_{2_i} t)$

	Task A (*n*=800)	**Task B** (*n*=720)	**Task C** (*n*=480)	**Task D** (*n*=800)
$\boldsymbol{A}$	[0.8, 0.5 0.7 2.0 3.0 3.0]	[0.8 0.4 0.8 1.0 0.7 0.3]	[0.4, 0.6 0.8 0.5 0.3 0.4]	[1.0 0.5 0.7 0.5 1.2 1.1
$\boldsymbol{\omega_1}$	[0.45 0.45 0.35 0.20 0.35 0.30]π	[0.47 0.33 0.40 0.53 0.67 0.47]π	[0.90 0.80 0.70 0.60 0.50 0.40]π	[0.44 0.36 0.28 0.36 0.24 0.36]π
$\boldsymbol{\omega_2}$	[0.25 0.50 0.50 0.20 0.40 0.25]π	[0.67 0.80 0.47 0.53 0.47 0.73]π	[0.80 0.70 0.70 0.50 0.30 0.60]π	[0.24 0.44 0.40 0.20 0.48 0.28]π

Acknowledgment

This work was partly supported by the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government (MEST) (NRF-M1AXA003-2011-0031425).

References

[1] C. H. An, C. G. Atkeson, J. M. Hollerbach, *Model-based control of a robot manipulator*. Cambridge: MA: MIT Press, 1988.

[2] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, *Robotics: Modeling, Planning and Control*. London: Springer, 2009, ch. 7.

[3] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*, Massachusetts Institute of Technology: MIT-Press, 2006.

[4] S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression: An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space," in *Proc. 16th Int. Conf. on Machine Learning (ICML)*, 2000, pp. 288-293.

[5] D. Nguyen-Tuong and J. Peters, "Model Learning for Robot Control: A Survey," Cognitive Processing, vol. 12, no. 4, pp.319-340, 2011.

[6] D. Nguyen-Tuong and J. Peters, "Computed Torque Control with Nonparametric Regression Models," in *Proc. the 2008 American Control Conference (ACC)*, 2008, pp. 212-217.

[7] J. Peters and S. Schaal, "Learning to Control in Operational Space,*" Int. Journal of Robotics Research*, vol. 27, no. 2, pp.197-212, 2008.

[8] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2010.

[9] B. Bócsi, L. Csató and J. Peters, "Alignment-based Transfer Learning for Robot Models", in *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, 2013, pp. 1-7.

[10] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu, "Text Classification without Negative Examples Revisit," IEEE *Trans. Knowledge and Data Enineering*, vol. 18, no. 1, pp. 6-20, 2006.

[11] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu, "Transfer Learning for WiFi-based Indoor Localization," *in Proc. Workshop on Transfer Learning for Complex Task of the 23rd Assoc. for the Advancement of Artificial Intelligence (AAAI) Conf. Artificial Intelligence*, July 2008.

[12] M. E. Taylor and P. Stone, "Cross-Domain Transfer for Reinforcement Learning," *in Proc. 24th Int. Conf. Machine Learning (ICML)*, 2007, pp. 699-707.

[13] D. Nguyen-Tuong, J. Peters, "Using Model Knowledge for Learning Inverse Dynamics," in *Proc. IEEE 16th Int. Conf. on Machine Learning (ICML)*, 2000, pp. 288-293.

[14] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24-32, 1996.

[15] M.T. Rosentein, Z. Marx, and L.P. Kaelbling, "To Transfer or Not to Transfer," In Proc. of Conf. Neural Information Processing Systems (NIPS'05) Workshop Inductive Transfer: 10 Years Later, 2005.