Knowledge Transfer Between Robots with Similar Dynamics for High-Accuracy Impromptu Trajectory Tracking

Siqi Zhou[1], Andriy Sarabakha[2], Erdal Kayacan[3], Mohamed K. Helwa[1], and Angela P. Schoellig[1]

***Abstract*— In this paper, we propose an online learning approach that enables the inverse dynamics model learned for a source robot to be transferred to a target robot (e.g., from one quadrotor to another quadrotor with different mass or aerodynamic properties). The goal is to leverage knowledge from the source robot such that the target robot achieves high-accuracy trajectory tracking on arbitrary trajectories *from the first attempt* with minimal data recollection and training. Most existing approaches for multi-robot knowledge transfer are based on post-analysis of datasets collected from both robots. In this work, we study the feasibility of *impromptu* transfer of models across robots by learning an error prediction module online. In particular, we analytically derive the form of the mapping to be learned by the online module for exact tracking, propose an approach for characterizing similarity between robots, and use these results to analyze the stability of the overall system. The proposed approach is illustrated in simulation and verified experimentally on two different quadrotors performing impromptu trajectory tracking tasks, where the quadrotors are required to accurately track arbitrary hand-drawn trajectories from the first attempt.**

I. Introduction

Machine learning techniques have been applied to many robot control problems with the goal of achieving high performance in the presence of uncertainties in the dynamics and the environment [1]. Due to the cost associated with data collection and training, approaches such as manifold alignment [2]–[4] and learning invariant features [5], [6] have been proposed to transfer knowledge between robots and thereby increase the efficiency of robot learning. In these approaches, datasets on a set of sample tasks are initially collected from both robots. They are then used for finding a mapping offline to transfer knowledge from a source robot to a target robot. This transferred knowledge is expected to speed up the training of the target robot and enhance its performance in untrained tasks [7].

In this paper, we consider the problem of impromptu trajectory tracking, in which robots are required to track arbitrary trajectories accurately from the first attempt [8]. Model-based techniques such as model predictive control (MPC) or the linear-quadratic regulator (LQR) can be used to solve tracking problems; however, applying these techniques to achieve high tracking performance can be difficult as they rely on sufficiently accurate dynamics models or can be time-consuming to tune. In [8], [9], we proposed a deep neural network (DNN)-based approach to enhance the tracking performance of black-box robot control systems. In particular, we showed that we can effectively enhance the tracking performance of a robot by training a DNN inverse dynamics module offline and then pre-cascading the module to the baseline system at test time. For example, on 30 arbitrary, unseen hand-drawn trajectories, the DNN-enhancement approach reduced the tracking error of a quadrotor by an average of 43% [8].

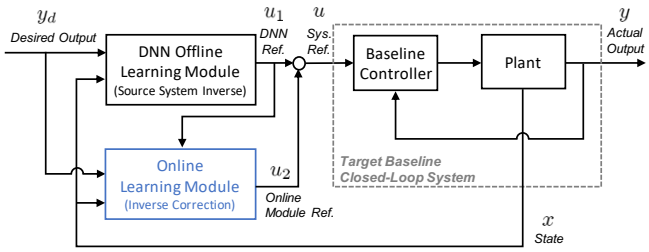


Fig. 1: Block diagram of the DNN-enhanced control architecture with online learning. The DNN module represents the inverse dynamics of a source system and is previously trained offline with a sufficiently rich dataset. During the testing phase, the DNN module is leveraged to enhance the tracking performance of a target system that shares some dynamic similarities with the source system. An online learning module (trained based on small sets of real-time data) further adjusts the reference generated by the DNN module to allow the target system to achieve high-accuracy tracking on arbitrary trajectories from the first attempt (i.e., impromptu tracking). A video of this work can be found here: http://tiny.cc/dnnTransfer

Motivated by recent work in transfer learning, in this work, we study the feasibility of leveraging the DNN model trained on one robot to enhance the performance of another robot in impromptu tracking tasks. In contrast to the existing approaches, where transfer mappings are usually found offline (e.g., [3], [5]), we propose an online learning approach (Fig. 1) that allows a target robot using the DNN module from a source robot to achieve high-accuracy tracking *impromptu* — i.e., without additional data collection and training on sample tasks. With the online learning approach, we aim to significantly reduce the data recollection and training time usually required for enhancing the target robot performance. In this work, we

(1) analytically derive the form of the mapping for the online module that allows the target system to achieve exact tracking,
(2) present first results on characterizing system similarity between source and target systems and how it relates

[1]Siqi Zhou, Mohamed K. Helwa, and Angela P. Schoellig are with the Dynamic Systems Lab (http://www.dynsyslab.org), Institute for Aerospace Studies, University of Toronto, Canada. The authors are also affiliated with the Vector Institute for Artificial Intelligence, Toronto. Mohamed K. Helwa is also with the Electrical Power and Machines Department, Cairo University, Egypt. Emails: siqi.zhou@robotics.utias.utoronto.ca, mohamed.helwa@robotics.utias.utoronto.ca, schoellig@utias.utoronto.ca

[2]Andriy Sarabakha is with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. Email: andriy001@e.ntu.edu.sg

[3]Erdal Kayacan is with the Department of Engineering, Aarhus University, Denmark. Email: erdal@eng.au.dk

to the stability of the proposed overall learning system given modeling uncertainties, and

(3) verify the effectiveness of the proposed approach in simulation and impromptu trajectory tracking experiments on quadrotors.

The paper is organized as follows: We start with a brief review of the related work (Sec. II) and provide some background on offline inverse learning (Sec. III). Then, we formulate the online learning problem (Sec. IV), discuss theoretical results (Sec. V), and illustrate the approach in simulation (Sec. VI) and in quadrotor experiments (Sec. VII). We conclude with a summary of the main results (Sec. VIII).

II. Related Work

The problem of knowledge transfer or transfer learning has been studied in different application domains (e.g., natural language processing [10], computer vision [11], and robot control [3]). The common goal is to leverage existing data to accelerate and improve subsequent learning processes such that the costs (and potential risks) associated with data recollection can be reduced [7], [12]. In robotics, two directions of knowledge transfer have been considered: *(i)* transfer across tasks and *(ii)* transfer across robots. The former typically considers the transfer of knowledge from a source task to a target task to be performed by a single robot (e.g., [13]–[15]), while the latter considers the transfer of knowledge from a source robot to a target robot (e.g., [2]–[6], [16]). In this paper, we will focus on the latter. We aim to transfer the inverse dynamics model trained on one robot to enhance the tracking performance of another robot. The transferred inverse dynamics model is expected to generalize to arbitrary trajectories [8], [9].

In the robot learning literature, and especially in reinforcement learning (RL), different approaches have been proposed to address the problem of knowledge transfer across different robots or domains. One of the approaches for cross-domain transfer is manifold alignment, where data from the source and target systems are collected for a set of sample tasks and are mapped to corresponding feature spaces (e.g., through dimensionality reduction) from which a transformation mapping between the source and target systems is found. This offline mapping can then be used to translate the policies trained on the source robot to the policies for the target robot [2], or map the data collected on the source robot to the target robot for model learning [3]. Extension hereto [4], [16] derive an optimal mapping for data transfer across robots from a control theory perspective. Other related work aims to learn and exploit a common feature space between the source and target robots while performing similar tasks [5], [6]. In [6], it is shown that the approach can effectively transfer control policies across different quadrotor platforms for autonomous navigation.

In addition to the above, there are a few other lines of relevant work involving knowledge transfer. One of them is sim-to-real [17], [18], where the low-cost data from a simulation is exploited for accelerating the training on physical robots. Moreover, in meta learning, the learning parameters are optimized for initializing subsequent learning [19]. In [20], modularity in learning has also been proposed to maximize the utility of learned models.

Although recent literature demonstrates the possibility of transferring knowledge across robots, we address two additional aspects in our paper. The first aspect is *impromptu* knowledge transfer without a-priori data collection on target systems. The second aspect is the impact of dynamic system similarity on the feasibility of knowledge transfer. An open question in the transfer learning literature is the issue of negative transfer (i.e., when the transfer adversely affects the target system) [7]. While researchers have investigated task similarity in the context of task transfer problems [21], discussions on system similarity for transferring knowledge across robots are rare. In this paper, we present theoretical results that associate system similarity to the feasibility of knowledge transfer across robots.

III. Background on Offline Inverse Learning

In this section, we provide more information about the DNN module (Fig. 1) to facilitate the discussions in the following sections. In [9], we considered a nonlinear closed-loop baseline system represented by

$$\begin{aligned} x(k+1) &= f\left(x(k)\right) + g\left(x(k)\right)u(k) \\ y(k) &= h\left(x(k)\right), \end{aligned} \tag{1}$$

where $k \in \mathbb{Z}_{\geq 0}$ is the discrete time index, $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}$ and $y \in \mathbb{R}$ are the input and output of the system, respectively, and $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are smooth functions. System (1) is said to have a relative degree r at a point (x_0, u_0) if $\frac{\partial}{\partial u} h \circ f^p(f(x(k)) + g(x(k))u(k)) = 0$ for $p = 0, ..., r-2$ for all points (x, u) in the neighbourhood of (x_0, u_0), and $\frac{\partial}{\partial u} h \circ f^{r-1}\big(f(x(k)) + g(x(k))u(k)\big) \neq 0$ at (x_0, u_0), where $(h \circ f)(x)$ is $h(f(x))$, and f^i is the i-th composition of function f with $f^0(x(t)) = x(t)$ and $f^i(x(t)) = f^{i-1} \circ (f(x(t)))$ [22]. For a system with a relative degree r, one may relate its input and output by $y(k+r) = h \circ f^{r-1}\big(f(x(k)) + g(x(k))u(k)\big)$. For many practical systems (e.g., manipulators), the output $y(k+r)$ can be written as an affine function in the input $u(k)$:

$$y(k+r) = \mathcal{F}\left(x(k)\right) + \mathcal{G}\left(x(k)\right)u(k), \tag{2}$$

where $\mathcal{F}\left(x(k)\right) = h \circ f^r(x(k))$ and $\mathcal{G}\left(x(k)\right) = \frac{\partial}{\partial u} h \circ f^{r-1}(f(x(k)) + g(x(k))u(k))$ [22], [23]. From Eqn. (2), one can show that the reference signal for exact tracking (i.e., $y(k+r) = y_d(k+r)$) is

$$u(k) = \frac{1}{\mathcal{G}\left(x(k)\right)} \left(y_d(k+r) - \mathcal{F}\left(x(k)\right)\right). \tag{3}$$

In more general cases, we can assume that the reference $u(k)$ for exact tracking is a nonlinear function of the state $x(k)$ and the future desired output $y_d(k+r)$. In [8], [9], we showed that, for an unknown, minimum-phase, nonlinear baseline system with a well-defined relative degree, we can train a DNN module that approximates the closed-loop inverse dynamics in Eqn. (3) and effectively enhances the tracking performance of the baseline system. In particular, in the

training phase of the DNN module, we construct a dataset with input $\{x(k), y(k+r)\}$ and output $\{u(k)\}$ based on the input-output response data from the baseline system. In the testing phase, the DNN module is pre-cascaded to the baseline system to adjust the reference $u(k)$ based on the current state $x(k)$ and desired output $y_d(k+r)$ (see Fig. 1).

Although we considered stable closed-loop baseline systems in [9], the results can be extended to that for stablizable open-loop plants. The approach in [9] decouples the problem of stabilization from the problem of improving tracking performance, which makes the overall learning-based approach more effective and less prone to instabilities.

IV. Problem Formulation

We consider the control architecture in Fig. 1 and study the knowledge transfer problem that allows the DNN module trained on a source robot system to enhance the impromptu tracking performance of a target robot system that has different dynamics. As in [9], the source and target robot systems are closed-loop systems whose dynamics can be represented by Eqns. (1) and (2). We assume that:

(A1) The source and target systems are input-to-state stable [24].

(A2) The source and target systems *(i)* have well-defined and the same relative degree, and *(ii)* are minimum phase.

(A3) The desired trajectory y_d is bounded, and a preview of $y_d(k+r)$ is available at time step k.

Note that (A1) and (A2) are necessary for safe operations and for applying the DNN inverse learning [9]. In (A2), we also assume that the source and target systems have the same relative degree to simplify the analysis. This condition holds, for instance, if the two robots have similar structures but different parameters (e.g., masses and dimensions). For (A3), the relative degree of a system is typically a small integer bounded by the system order, and a preview of r time steps of the desired trajectory can typically be achieved by online and offline trajectory generation algorithms.

V. Theoretical Results

In this section, we consider the control architecture in Fig. 1 and provide theoretical results related to the knowledge transfer problem. We denote u_1 as the reference from the DNN module trained on the source system and u_2 as the reference from the online learning module. The overall reference to the target baseline system $u(k)$ is given by

$$u(k) = u_1(k) + u_2(k). \tag{4}$$

Below we derive an expression of $u_2(k)$ for achieving exact tracking in Sec. V-A, propose a characterization of system similarity in Sec. V-B, and analyze the stability of the overall system in the presence of uncertainties in Sec. V-C.

A. Reference Adaptation for Exact Tracking

In this subsection, we derive an expression for $u_2(k)$ such that $u(k)$ achieves exact tracking $y(k+r) = y_d(k+r)$, where y and y_d are the desired and actual outputs of the target system, and r is the system relative degree.

A common approach for high-accuracy trajectory tracking is to adapt the reference input of a nominal controller based on the observed tracking errors. For instance, in PD-type iterative learning control (ILC), proportional and derivative tracking error terms are added to the reference in each iteration to improve the tracking performance over a sequence of trials [25]. In distal teacher inverse dynamics learning, the tracking error is proposed as the cost function for updating the weights of a neural-network-based controller online to achieve improved tracking [26]. In this work, we similarly consider an online learning approach that adapts the reference of the DNN module $u_1(k)$ based on the tracking error. In particular, we justify below that the reference $u_2(k)$ can be approximated by

$$u_2(k) = \alpha\, e_p(k+r), \tag{5}$$

where α is an adaptation gain, and $e_p(k+r)$ is a prediction of the tracking error r time steps ahead.

We consider a nonlinear target system (1), (2):

$$y(k+r) = \mathcal{F}_t\left(x(k)\right) + \mathcal{G}_t\left(x(k)\right) u(k), \tag{6}$$

where $\mathcal{F}_t\left(x(k)\right) = h_t \circ f_t^r(x(k))$ and $\mathcal{G}_t\left(x(k)\right) = \frac{\partial}{\partial u} h_t \circ f_t^{r-1}(f_t(x(k)) + g_t(x(k))u(k))$, and $f_t(\cdot)$, $g_t(\cdot)$, and $h_t(\cdot)$ are the corresponding nonlinear functions in Eqn. (1). In addition to the target system, we consider a source system, which the DNN module is trained on. This system is similarly represented in the form of Eqn. (2). As discussed in [9], the underlying function approximated by the DNN is

$$u_1(k) = \frac{1}{\mathcal{G}_s\left(x(k)\right)} \left(y_d(k+r) - \mathcal{F}_s\left(x(k)\right)\right), \tag{7}$$

where $\mathcal{F}_s\left(x(k)\right)$ and $\mathcal{G}_s\left(x(k)\right)$ are defined analogously to those of the target system. By substituting Eqns. (4) and (7) into Eqn. (6), one can see that the ideal reference $u_2(k)$ for achieving exact tracking is

$$u_2(k) = \alpha^* e_p^*(k+r), \tag{8}$$

where $\alpha^* = \frac{1}{\mathcal{G}_t(x(k))}$ and

$$e_p^*(k+r) = y_d(k+r) - \mathcal{F}_t\left(x(k)\right) - \mathcal{G}_t\left(x(k)\right) u_1(k). \tag{9}$$

Insight 1. Ideal Mapping for Exact Tracking. In order to achieve exact tracking, the online learning module should predict the tracking error of the target system that would result from applying $u_1(k)$. The predicted error is scaled by a gain $\alpha^* = \frac{1}{\mathcal{G}_t(x(k))}$, where $\mathcal{G}_t(x(k)) = \frac{\partial y(k+r)}{\partial u(k)}$.

The error prediction in Eqn. (9) depends on the current state $x(k)$, the reference $u_1(k)$ from the DNN module, and the future desired output $y_d(k+r)$. When the dynamics of the source and the target systems are not known, one may use supervised learning to train a model online to approximate Eqn. (9). We present a general approach for training this online model in Remark 1.

Remark 1. Online Learning for Error Prediction. For training an online model to approximate Eqn. (9), at each time step k, one may construct a dataset with paired inputs $\{x(p-r), u(p-r), y_d(p)\}$ and outputs $\{y_d(p) - y(p)\}$ over the past N time steps $p = k-N, ..., k$. The error $e_p(k+r)$ can then be predicted using the online model with input $\mathcal{I} = [x(k), u_1(k), y_d(k+r)]$.

Given the predicted error $e_p(k+r)$, another component to be determined for computing $u_2(k)$ is the gain α. With an online model $F\left(x(k), u_1(k), y_d(k+r)\right)$ approximating Eqn. (9), it can be shown that α^* can be obtained from $\hat{\alpha}^* = -\left(\partial F/\partial u_1\right)^{-1}$. In practice, due to noise in the systems, the online estimation of α^* can be non-trivial. In Sec. V-C, we provide an analysis to examine the stability of the overall system when α^* is approximated by a constant and also when the estimation of $e_p^*(k+r)$ by the online model is inexact.

B. System Similarity

The concept of task similarity has been introduced in the RL literature to address the issue of negative knowledge transfer in task transfer learning problems [21]. In this subsection, we propose a characterization of system similarity for impromptu knowledge transfer problems, where an inverse module is transferred across two robot systems.

We consider two systems are similar if at any given state $x(k)$, the application of an input $u(k)$ to the systems results in similar outputs $y(k+r)$ [27]. For the similarity discussion, we assume linear or linearized source and target systems to simplify our analysis:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{aligned} \tag{10}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}$ is the input, $y \in \mathbb{R}$ is the output, and (A, B, C) are constant matrices. It can be shown that the input and output of system (10) are related by

$$y(k+r) = \mathcal{A}x(k) + \mathcal{B}u(k). \tag{11}$$

where $\mathcal{A} = CA^r$ and $\mathcal{B} = CA^{r-1}B$, and r is the relative degree of system (10). From Eqn. (11), the input-output relationship is fully characterized by $\mathcal{A}$ and $\mathcal{B}$, which can be thought as the state-to-output gain vector and the input-to-output gain, respectively. Based on the relationship in Eqn. (11), we define a vector S to characterize the similarity of the source and target systems:

$$S = \begin{bmatrix} S_1 & S_2 \end{bmatrix}, \tag{12}$$

where $S_1 = 1 - \frac{\mathcal{B}_t}{\mathcal{B}_s}$, $S_2 = \mathcal{A}_t - \frac{\mathcal{B}_t}{\mathcal{B}_s}\mathcal{A}_s$, and the subscripts s and t denote the source and the target system. The terms S_1 and S_2, respectively, characterize the differences in the input-to-output gain and state-to-output gain vector of the source and target systems. Note that $S = 0$ if and only if $\mathcal{A}_t = \mathcal{A}_s$ and $\mathcal{B}_t = \mathcal{B}_s$ (i.e., the state-to-output and input-to-output gains of the systems are identical).

C. Stability in the Presence of Uncertainties

In this subsection, we use the concept of system similarity and analyze the stability of the target system when the gain α^* is approximated by a constant α and the prediction of the future error $e_p^*(k+r)$ is not exact. We focus on system (10) and make the following assumptions:

(A4) The output of the offline DNN $u_1(k)$ corresponds to the inverse of the source system $u_1(k) = \frac{1}{\mathcal{B}_s}\left(y_d(k+r) - \mathcal{A}_s x(k)\right)$, where $\mathcal{A}_s$ and $\mathcal{B}_s$ are the gains of the source system, and $x(k)$ and $y_d(k+r)$ are the state and desired output of the target system.

(A5) The error in the prediction $\Lambda = e_p^*(k+r) - e_p(k+r)$ can be bounded as follows: $\Lambda \leq \beta_1||y_d(k+r)|| + \beta_2||x(k)|| + \beta_3$, where β_1, β_2, and β_3 are positive constants, and $||\cdot||$ is the Euclidean norm.

In addition, by (A1), the target system is input-to-state stable. It can be shown that the state of system (10) can be bounded as follows: $||x||_\infty \leq L_1||u||_\infty + L_2||x_0||$, where $||x||_\infty = \sup_k\{||x(k)||\}$, $||u||_\infty = \sup_k\{||u(k)||\}$, and L_1 and L_2 are positive constants.

Lemma 1. Stability. Consider a target system represented by Eqn. (10) and the control architecture in Fig. 1, where the reference of the online learning module $u_2(k)$ has the form of Eqn. (5). Under (A1), (A4), and (A5), the overall system is bounded-input-bounded-state (BIBS) stable if

$$|\alpha|\left(||S_2|| + \beta_2\right) < \frac{\beta_4}{L_1}, \tag{13}$$

where $\beta_4 = 1 - L_1\left|\left|\frac{\mathcal{A}_s}{\mathcal{B}_s}\right|\right|$.

Proof. At a time step k, the output of the online learning module is $u_2(k) = \alpha\, e_p(k+r)$, where α is a constant gain and $e_p(k+r)$ is the predicted tracking error. The adjusted reference $u(k)$ sent to the target baseline system is $u(k) = u_1(k) + \alpha\, e_p(k+r)$, where $u_1(k)$ is the output of the offline DNN module. By (A4) and (A5), we can write $u(k)$ as

$$u(k) = \frac{1}{\mathcal{B}_s}\left(y_d(k+r) - \mathcal{A}_s x(k)\right) + \alpha\left(e_p^*(k+r) - \Lambda\right). \tag{14}$$

For a target system represented by Eqn. (10), $e_p^*(k+r)$ in Eqn. (9) can be written as $e_p^*(k+r) = y_d(k+r) - \mathcal{A}_t x(k) - \mathcal{B}_t u_1(k) = y_d(k+r) - \mathcal{A}_t x(k) - \frac{\mathcal{B}_t}{\mathcal{B}_s}\left(y_d(k+r) - \mathcal{A}_s x(k)\right)$. By substituting the expression of $e_p^*(k+r)$ into Eqn. (14), we obtain $u(k) = \left(\frac{1}{\mathcal{B}_s} + \alpha S_1\right) y_d(k+r) - \left(\frac{\mathcal{A}_s}{\mathcal{B}_s} + \alpha S_2\right) x(k) - \alpha\Lambda$. Moreover, by (A1) and (A5), we can relate $||x||_\infty$ to $||y_d||_\infty = \sup_k\{||y_d(k)||\}$ by the following inequality:

$$\begin{aligned} ||x||_\infty \leq L_1 \Bigg(&\left(\left|\frac{1}{\mathcal{B}_s}\right| + |\alpha|\,|S_1| + \beta_1|\alpha|\right)||y_d||_\infty \\ &+ \left(\left|\left|\frac{\mathcal{A}_s}{\mathcal{B}_s}\right|\right| + |\alpha|\,||S_2|| + \beta_2|\alpha|\right)||x||_\infty\Bigg) \\ &+ L_1\beta_3|\alpha| + L_2||x_0||. \end{aligned} \tag{15}$$

From Eqn. (15), if $1 - L_1\left(\left|\left|\frac{\mathcal{A}_s}{\mathcal{B}_s}\right|\right| + |\alpha|\,||S_2|| + \beta_2|\alpha|\right) > 0$, or equivalently $|\alpha|\left(||S_2|| + \beta_2\right) < \frac{\beta_4}{L_1}$, then the state of the system can be bounded as follows: $||x||_\infty \leq \frac{L_1\left(\left|\frac{1}{\mathcal{B}_s}\right| + |\alpha||S_1| + \beta_1|\alpha|\right)||y_d||_\infty + L_1\beta_3|\alpha| + L_2||x_0||}{1 - L_1\left(\left|\left|\frac{\mathcal{A}_s}{\mathcal{B}_s}\right|\right| + |\alpha|||S_2|| + \beta_2|\alpha|\right)}$. Now, if y_d and hence $||y_d||_\infty$ are bounded, then the system state is bounded,

and the overall system is BIBS stable. □

Recall that, in Eqn. (13), α is the gain of the online learning module, S_2 characterizes the similarity between the two systems, β_1 is associated with the uncertainty in the error prediction, and L_1 can be thought of as a characterization of the aggressiveness of the target system. The condition in Eqn. (13) can be interpreted for two scenarios: (*i*) when $|\alpha| = 0$ (i.e., the online module is inactive) and *(ii)* when $|\alpha| \neq 0$ (i.e., the online module is active). In scenario *(i)*, the condition in Eqn. (13) reduces to $L_1 < \frac{1}{||\mathcal{A}_s/\mathcal{B}_s||}$, which can be interpreted as an upper bound on the relative aggressiveness of the source and target systems. When this condition is satisfied, the target system with the source system DNN module is stable. In scenario *(ii)*, when the online learning module is active, the condition in Eqn. (13) implies that if the source and target systems are more similar, that is $||S_2||$ is closer to 0, then there will be a greater margin for selecting α and higher tolerance for having uncertainties in the online prediction model. Moreover, based on the condition in Eqn. (13), one may use probabilistic learning techniques to estimate the uncertainties in the predicted error $e_p(k+r)$ and calculate an upper bound on the magnitude of the fixed gain α for stability.

Remark 2. Generalization to Nonlinear Systems. For nonlinear systems (1) with inputs and outputs related by Eqn. (2), one can relate the outputs of the source and target systems by $y_t(k+r) = \vartheta_1(x(k))\, y_s(k+r) + \vartheta_2(x(k))$, where $\vartheta_1(x(k)) = \frac{\mathcal{G}_t(x(k))}{\mathcal{G}_s(x(k))}$ and $\vartheta_2(x(k)) = \mathcal{F}_t(x(k)) - \frac{\mathcal{G}_t(x(k))}{\mathcal{G}_s(x(k))}\mathcal{F}_s(x(k))$. The relation between $y_t(k+r)$ and $y_s(k+r)$ can be used as an alternative for characterizing the similarity for the nonlinear systems. It is left for future work to perform a similar stability analysis for the nonlinear case.

VI. Simulation Illustration

In this section, we illustrate the proposed online learning approach with a simulation example. In [9], we considered a minimum phase closed-loop baseline system represented by

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 1 \\ -0.15 & 0.8 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} -0.2 & 1 \end{bmatrix} x(k), \end{aligned} \tag{16}$$

and showed that a DNN module (Sec. III) can be designed to enable the system to achieve exact tracking on untrained trajectories. In the following simulation study, we consider system (16) as the source system and leverage its offline DNN module to enhance the tracking performance of a target system that is represented by

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 1 \\ -0.24 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} -0.1 & 1 \end{bmatrix} x(k). \end{aligned} \tag{17}$$

Note that the source system (16) and the target system (17) are minimum phase and have relative degrees of 1. The source system has two poles at $\{0.3, 0.5\}$ and a zero at 0.2, while the target system has two poles at $\{0.4, 0.6\}$ and a zero at 0.1. When implementing the learning modules, we assume that the systems are black boxes, and we rely on only their input-output data and basic properties (e.g., relative degree).

A. Learning Modules

1) Offline Learning of Inverse Module: The offline inverse module is trained on a source system (e.g., a system that is similar to the target system or a simulator), from which abundant data has been collected. The collected data can often be compactly represented by parametric regression techniques [8], [9]. For the source system (16), we adopt the DNN module from [9] and transfer this inverse module to enhance the target system (17) with the proposed online learning approach. The DNN module of the source system is a 3-layer feedforward network with 20 hyperbolic tangent neurons in each hidden layer. The input and output of the DNN module are $\mathcal{I} = [x(k), y_d(k+1)]$ and $\mathcal{O} = u_1(k)$. The training dataset is constructed from the source system's response on 25 sinusoidal trajectories with different combinations of frequencies and amplitudes; Matlab's Neural Network Toolbox is used to train the DNN model parameters offline. More details of the DNN training can be found in [9].

2) Error Prediction with Online Learning: The online error prediction module is a local model trained on a small dataset constructed from the latest observations of the target system. The objective of incorporating the online module is to achieve fast adaptation to the dynamic differences between the source and target systems. In the simulation, a Gaussian process (GP) regression model is utilized for learning the error prediction module online. Based on Remark 1, the input and output of the online module are selected to be $\mathcal{I} = [x(k), u_1(k), y_d(k+1)]$ and $\mathcal{O} = e_p(k+1)$, respectively. At each time step k, a fixed-sized training dataset is constructed based on the latest 15 observations; in particular, the input and output are $\{(x(p-1), u(p-1), y_d(p))\}$ and $\{y_d(p) - y(p)\}$ for $p = k-15, ..., k$. For the simulation, the GP model uses the squared-exponential kernel $K(\xi, \xi') = \sigma_1^2 \exp\left(-\frac{1}{2}\sum_i \frac{(\xi_i - \xi_i')^2}{l_i^2}\right)$ and polynomial explicit basis functions $\{1, \xi_i, \xi_i^2\}$, where ξ denotes the input to the module and ξ_i denotes the i-th component of ξ, l_i is the length scale associated with the input dimension ξ_i, and σ_1^2 is the prior variance [28]. The length scales l_i are identical for all input dimensions in the simulation; the hyperparameters of the kernel function and the coefficients of the basis functions are optimized online with Matlab's Gaussian Process Regression toolbox. The gain α^* is estimated based on the online error prediction module as $\hat{\alpha}^* = -\left(\partial F_{\text{gpr}}/\partial u_1\right)^{-1}$, where F_{gpr} denotes the function represented by the GP regression model.

B. Simulation Results

Figures 2 and 3 show the performance of the learning modules and the target system on a test trajectory $y_d(t) = \sin\left(\frac{2\pi}{8}t\right) + \cos\left(\frac{2\pi}{16}t\right) - 1$, where $t = 1.5 \times 10^{-3}k$ is the continuous-time variable. This test trajectory is not previously used in the training of the offline learning module.

Figure 2 compares the predicted error from the online module and the analytical error prediction of the target system computed based Eqn. (9). It can be seen that the online module designed based on Remark 1 is able to accurately predict the error of the target system that would result from applying the reference u_1 alone. On the test trajectory, the root-mean-square (RMS) error of the online module prediction is approximately 2.9×10^{-7}.

Figure 3 shows the outputs of the target system when *(i)* the baseline controller is applied (grey), *(ii)* the baseline system is enhanced by the offline module alone (green), and *(iii)* the baseline system is enhanced by both the online and the offline modules (blue). As compared to the baseline system, the offline module alone reduces the RMS tracking error of the target system from 3.97 to 0.44. The online module further reduces the RMS tracking error to 9×10^{-5}. Applying the offline and the online learning modules jointly allows the target system to achieve approximately exact tracking on a test trajectory that is not seen by the source or the target system a-priori.

VII. Quadrotor Experiments

With impromptu tracking of hand-drawn trajectories as the benchmark problem [8], we illustrate the proposed online learning approach for transferring the DNN module trained on a source quadrotor system, the Parrot ARDrone 2.0, to a target quadrotor system, the Parrot Bebop 2. A demo video can be found here: http://tiny.cc/dnnTransfer

A. Experiment Setup

In [8] and [9], with the ARDrone as the testing platform, it is shown that a DNN module trained offline can effectively enhance the impromptu tracking performance of the quadrotor on arbitrary hand-drawn trajectories. In this work, we leverage the DNN module trained on the ARDrone to enhance the impromptu tracking performance of the Bebop and further apply the proposed online learning approach (Remark 1) to achieve high-accuracy tracking.

1) Control Objective and Baseline Control System: The dynamics of a quadrotor vehicle can be characterized by 12 states: translational positions $\mathbf{p} = (x, y, z)$, translational velocities $\mathbf{v} = (\dot{x}, \dot{y}, \dot{z})$, roll-pitch-yaw angles $\boldsymbol{\theta} = (\phi, \theta, \psi)$, and rotational velocities $\boldsymbol{\omega} = (p, q, r)$. The objective is to design a control system such that the position of the quadrotor $\mathbf{p}_a$ tracks desired trajectories $\mathbf{p}_d$ generated from arbitrary hand drawings. In this work, we use the RMS error as the measure for evaluating tracking performance.

The baseline control systems of the quadrotor platforms, the ARDrone and the Bebop, have an offboard position controller running at 70 Hz and an onboard attitude controller running at 200 Hz. The offboard position controller receives the reference position $\mathbf{p}_r$ and reference velocity $\mathbf{v}_r$, and computes attitude commands ϕ_{cmd} and θ_{cmd}, yaw rate command r_{cmd}, and z-velocity command $\dot{z}_{\text{cmd}}$. The onboard attitude controller receives the commands from the offboard controller, and computes the desired thrusts of the four motors of the vehicle. In the experiments, we apply the

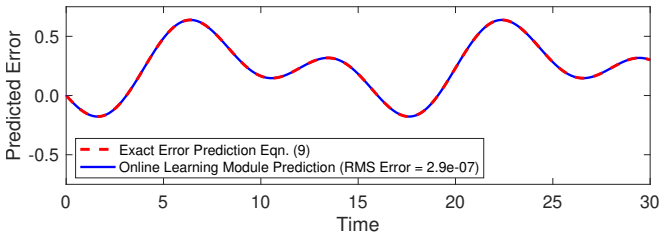


Fig. 2: A plot of the error prediction from the online learning module. The error predicted by an online module trained based on Remark 1 (blue) coincides with the exact error prediction computed based on Eqn. 9 (red).

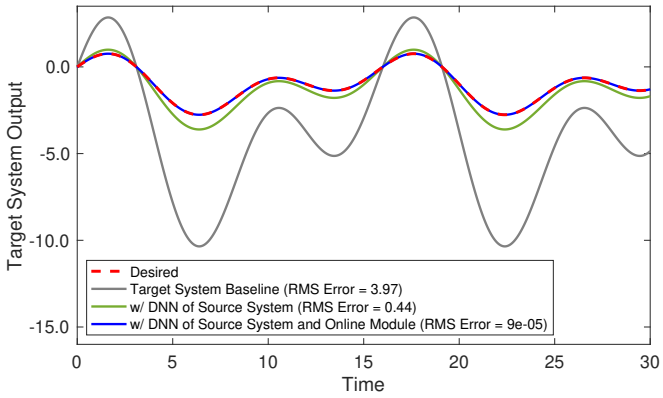


Fig. 3: Plots of the target system output when *(i)* the baseline controller (grey), *(ii)* the baseline controller with the offline learning module (green), and *(iii)* the baseline controller with both the offline and online learning modules (blue) are applied. Due to system similarity, the offline learning module (trained on the source system) significantly reduces the tracking error of the target system. With the further incorporation of the online learning module, exact tracking is approximately achieved.

offline and online learning modules to enhance the tracking performance of the baseline controller of the Bebop (the target system). In the design of the learning modules, we assume that the high-level dynamics of the ARDrone and the Bebop are decoupled in the x, y, and z directions.

2) DNN Module Trained on ARDrone (Source System): In [8], [9], a DNN module is trained offline to approximate the inverse of the ARDrone baseline system dynamics. Based on the theoretical insights in [9], the input and output of the DNN module are determined to be $\mathcal{I}_1 = [x_d(k+4) - x_a(k), y_d(k+4) - y_a(k), z_d(k+3) - z_d(k), \dot{x}_d(k+3) - \dot{x}_a(k), \dot{y}_d(k+3) - \dot{y}_a(k), \dot{z}_d(k+2) - \dot{z}_a(k), \boldsymbol{\theta}_a(k), \boldsymbol{\omega}_a(k)]$ and $\mathcal{O}_1 = [\mathbf{p}_r(k) - \mathbf{p}_a(k), \mathbf{v}_r(k) - \mathbf{v}_a(k)]$. The DNN module consists of fully-connected feedforward networks with 4 hidden layers of 128 rectified linear units (ReLU). The training dataset of the DNN module is constructed from the ARDrone baseline system response on a 400-second, 3-dimensional sinusoidal trajectory. At a sampling rate of 7 Hz, approximately 2,800 pairs of data points are collected for training. The DNN module is implemented using Tensorflow in Python. Further details of the DNN module implementation can be found in [8], [9]. As shown in [8], for 30 hand-drawn test trajectories, this offline DNN module is able to reduce the impromptu tracking error of the ARDrone baseline system by 43% on average.

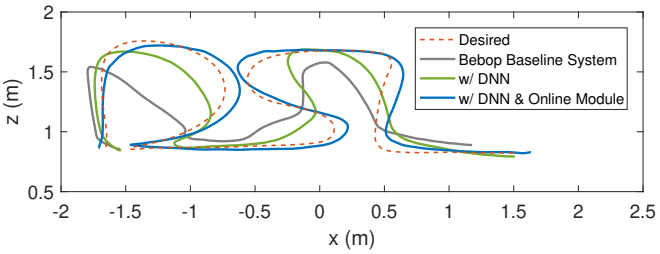
3) Online Learning for Bebop (Target System): Based on Remark 1, the input and output of the online learning module

are $\mathcal{I}_2 = [\mathbf{p}_a(k), \mathbf{v}_a(k), \boldsymbol{\theta}_a(k), \boldsymbol{\omega}_a(k), \mathbf{p}_r(k), \mathbf{v}_r(k), x_d(k+4), y_d(k+4), z_d(k+3), \dot{x}_d(k+3), \dot{y}_d(k+3), \dot{z}_d(k+2)]$ and $\mathcal{O}_2 = [x_e(k+4), y_e(k+4), z_e(k+3), \dot{x}_e(k+3), \dot{y}_e(k+3), \dot{z}_e(k+2)]$, where $(\cdot)_e$ denotes the predicted position and velocity tracking errors of the Bebop system when the offline DNN trained on the ARDrone system is used. In the experiment, in order to make online learning more efficient, instead of predicting the position and velocity errors directly, we train a GP model to predict the position of the Bebop $\mathbf{p}_a(k+r) = [x_a(k+4), y_a(k+4), z_a(k+3)]$ and computes the predicted error by subtracting the predicted position from future desired position $\mathbf{p}_d(k+r) - \mathbf{p}_a(k+r)$, where $\mathbf{p}_d(k+r) = [x_d(k+4), y_d(k+4), z_d(k+3)]$. The predicted position errors are used to compute the corrections for the position components; the velocity reference corrections are numerically approximated with a first-order finite difference scheme. For the experiments, the online learning module is implemented by using the GPy library in Python. We use a standard squared-exponential kernel with a fixed length scale l for all input dimensions, prior variance σ_1^2, and zero mean Gaussian measurement noise with variance σ_2^2 [28]. At each time step k, the most recent 40 observations are used for constructing the training dataset. The hyperparameters of the GP model are $l = 20$, $\sigma_1^2 = 1$, and $\sigma_2^2 = 2 \times 10^{-5}$; these values are manually tuned a-priori for our experimental setup. If computational resources permit, we expect finer tuning of the hyperparameters online would lead to lower generalization errors and better tracking performance. Due to the measurement noise in the experiment, instead of estimating the parameter α online, we used constant gains $\alpha = (5, 5, 0.5)$ for the x, y, and z directions.

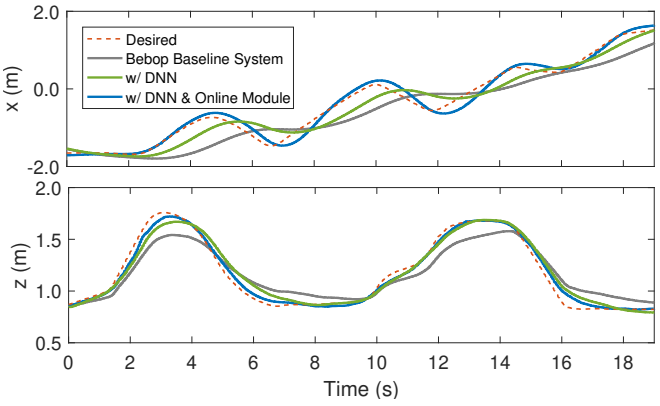
B. Experiment Results

Figure 4 compares the tracking performance of three control strategies on the Bebop on one of the test hand-drawn trajectories. When comparing the performance of the Bebop system enhanced by the ARDrone DNN (green) and the performance of the Bebop baseline system (grey), the ARDrone DNN reduces the delay and the amplitude errors in the Bebop tracking response. Along this particular trajectory, the DNN module alone reduces the RMS tracking error of the Bebop from approximately 0.42 m to 0.26 m. When further comparing with the performance of the DNN-enhanced system with the addition of the online learning module (blue), the tracking of the Bebop, especially in the x-direction, is brought close to the desired trajectory. With the online learning module, the RMS tracking error is reduced to approximately 0.14 m. Note that, from the plots in Fig. 4, when the online learning module is applied, there are small overshoots at the locations with larger curvatures. The overshoots may be reduced with online tuning of the GP hyperparameters and online estimations of the α parameters.

Figure 5 summarizes the performance errors of the three control strategies on 10 hand-drawn trajectories. When compared with the Bebop baseline system performance (grey), the direct application of the transferred DNN module (green) reduces the RMS tracking error of the Bebop baseline system



(a) Path of the target system in the x-z plane.



(b) Position trajectories of the target system.

Fig. 4: Comparison of three control strategies for the Bebop target system: The RMS error is 0.42 m for the Bebop baseline system (grey), 0.26 m for the baseline system enhanced by the ARDrone DNN (green), and 0.14 m for the baseline system further enhanced by the online learning module (blue).

by an average of 46%. With the addition of the online learning module (blue), an average of 74% RMS tracking error reduction is achieved. Two additional sets of results are included for comparison: *(i)* the performance of the ARDrone enhanced by the DNN module trained on the ARDrone system (yellow) and *(ii)* the performance of the Bebop enhanced by a DNN module trained on the Bebop system (light blue). Without requiring further data collection and offline training, the inclusion of the online learning module effectively reduces the RMS tracking error of the Bebop to values that are comparable to those of the cases where the quadrotors are enhanced by their own offline DNN modules. These results demonstrate the efficiency of the proposed online learning module to leverage past experience and reduce data re-collection and training.

VIII. Conclusions and Future Work

In this paper, we consider the impromptu tracking problem and propose an online learning approach to efficiently transfer a DNN module trained on a source robot system to a target robot system. In the theoretical analysis, we derive an expression of the online module for achieving exact tracking. Then, based on a linear system formulation, we propose an approach for characterizing system similarity and provide insights on the impact of the system similarity on the stability of the overall system in the knowledge transfer problem. We verify our approach experimentally by applying the proposed online learning approach to transfer a DNN inverse dynamics module across two quadrotor platforms (Parrot ARDrone and Bebop). On 10 arbitrary hand-drawn trajectories, the DNN module of the source system reduces

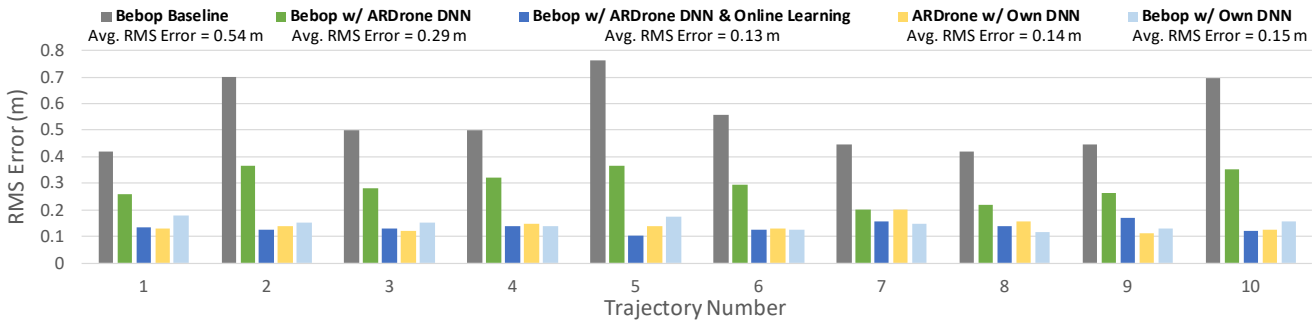


Fig. 5: Tracking performance of the target system (Bebop) on 10 hand-drawn trajectories. The ARDrone DNN module alone (green) and the ARDrone DNN module with the online learning module (blue) reduce the tracking error of the Bebop baseline system (grey) by 46% and 74% on average, respectively. With the proposed online learning approach, the average RMS error of the Bebop (blue) is comparable to cases where the ARDrone and the Bebop are enhanced by their own offline DNN modules (yellow and light blue).

the tracking error of the target system by an average of 46%. The incorporation of the online module further reduces the tracking error and leads to an average of 74% error reduction. These experimental results show that the proposed online learning and knowledge transfer approach can efficaciously circumvent data recollection on the target robot, and thus, the costs and risks associated with training new robots to achieve higher performance in impromptu tasks.

Potential future work includes extending the theoretical formulation to multi-input-multi-output (MIMO) systems, extending the stability analysis to nonlinear systems, and testing the approach in different outdoor conditions and on different robot platforms.

REFERENCES

[1] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12(4), pp. 319–340, 2011.

[2] H. B. Ammar, E. Eaton, P. Ruvolo, and M. Taylor, "Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2015.

[3] B. Bócsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *Proc. of the Intl. Joint Conf. on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[4] M. K. Helwa and A. P. Schoellig, "Multi-robot transfer learning: A dynamical system perspective," in *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 4702–4708.

[5] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *Proc. of the Intl. Conf. on Learning Representations*, 2017.

[6] S. Daftry, J. A. Bagnell, and M. Hebert, "Learning transferable policies for monocular reactive MAV control," in *Proc. of the Intl. Symposium on Experimental Robotics*. Springer, 2016, pp. 3–11.

[7] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research (JMLR)*, vol. 10, pp. 1633–1685, dec 2009.

[8] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5183–5189.

[9] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2017, pp. 5201–5207.

[10] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proc. of the Association for Computational Linguistics Conf. on Empirical Methods in Natural Language Processing*, 2006, pp. 120–128.

[11] Z. Wang, Y. Song, and C. Zhang, "Transferred dimensionality reduction," in *Machine Learning and Knowledge Discovery in Databases*, W. Daelemans, B. Goethals, and K. Morik, Eds., 2008, pp. 550–565.

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

[13] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 4019–4026.

[14] K. Pereida, D. Kooijman, R. R. P. R. Duivenvoorden, and A. P. Schoellig, "Transfer learning for high-precision trajectory tracking through L1 adaptive feedback and iterative learning," *Intl. Journal of Adaptive Control and Signal Processing*, 2018.

[15] M. Hamer, M. Waibel, and R. D'Andrea, "Knowledge transfer for high-performance quadrocopter maneuvers," in *Proc. of the IEEE Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 1714–1719.

[16] K. Pereida, M. K. Helwa, and A. P. Schoellig, "Data-efficient multi-robot, multi-task transfer learning for trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 3(2), pp. 1260–1267, 2018.

[17] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 1557–1563.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Prof. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[19] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Proc. of Machine Learning Research (PMLR)*, vol. 78, 2017, pp. 357–368.

[20] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 2169–2176.

[21] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey," in *Reinforcement Learning*. Springer, 2012, pp. 143–173.

[22] T.-J. Jang, H.-S. Ahn, and C.-H. Choi, "Iterative learning control for discrete-time nonlinear systems," *Intl. Journal of Systems Science*, vol. 25(7), pp. 1179–1189, 1994.

[23] M. Sun and D. Wang, "Analysis of nonlinear discrete-time systems with higher-order iterative learning control," *Dynamics and Control*, vol. 11(1), pp. 81–96, 2001.

[24] E. D. Sontag, "Input to state stability: Basic concepts and results," in *Nonlinear and optimal control theory*. Springer, 2008, pp. 163–220.

[25] A. Hock and A. P. Schoellig, "Distributed iterative learning control for a team of quadrotors," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2016, pp. 4640–4646.

[26] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, vol. 16(3), pp. 307–354, 1992.

[27] M. Whorton, L. Yang, and R. Hall, *Similarity Metrics for Closed Loop Dynamic Systems*, ser. AIAA Guidance, Navigation and Control Conf. and Exhibit. American Inst. of Aeronautics and Astronautics, 2008.

[28] C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.