Data-Efficient Multi-Robot, Multi-Task Transfer Learning for Trajectory Tracking

Karime Pereida, Mohamed K. Helwa and Angela P. Schoellig

Abstract— Transfer learning has the potential to reduce the burden of data collection and to decrease the unavoidable risks of the training phase. In this paper, we introduce a multi-robot, multi-task transfer learning framework that allows a system to complete a task by learning from a few demonstrations of another task executed on another system. We focus on the trajectory tracking problem where each trajectory represents a different task, since many robotic tasks can be described as a trajectory tracking problem. The proposed, *multi-robot* transfer learning framework is based on a combined $\mathcal{L}_1$ adaptive control and iterative learning control approach. The key idea is that the adaptive controller forces dynamically different systems to behave as a specified reference model. The proposed *multi-task* transfer learning framework uses theoretical control results (e.g., the concept of vector relative degree) to learn a map from desired trajectories to the inputs that make the system track these trajectories with high accuracy. This map is used to calculate the inputs for a new, unseen trajectory. Experimental results using two different quadrotor platforms and six different trajectories show that, on average, the proposed framework reduces the first-iteration tracking error by 74% when information from tracking a different, single trajectory on a different quadrotor is utilized.

I. INTRODUCTION

This work aims to design a learning-based control method that guarantees high overall performance of robots carrying out complex tasks in changing environments. These control methods do not require the exact system model, and they are robust against model uncertainties, unknown disturbances, and changing dynamics. This is in contrast to traditional, model-based controllers, where small changes in the conditions may significantly deteriorate the controller performance and cause instability (see [1], [2] and [3]).

Training robots to operate in changing environments is complex and time-consuming. Transfer learning reduces the unavoidable risks of the training phase and the time needed to train each robot on each single task. In this paper, we develop a multi-robot, multi-task transfer learning framework that
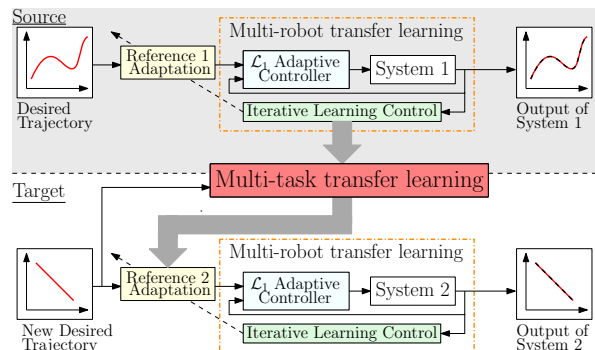
Fig. 1. The proposed multi-robot, multi-task architecture. The *multi-robot* transfer learning framework is composed of two methods *(i)* an iterative learning control (ILC) module to learn an input such that the output tracks a desired output signal and *(ii)* an $\mathcal{L}_1$ adaptive controller to force different systems to behave in the same repeatable, predefined way. Hence, trajectories learned on a source system can directly be transferred to a target system. The *multi-task* transfer learning framework learns a map from a desired trajectory to the inputs that make the system track it accurately. When a new trajectory is encountered, the learned map is used to calculate the inputs for the new trajectory.

allows a system to complete a task by learning from a few demonstrations of another task executed on another system, see Fig. 1. We focus on trajectory tracking as many robotic tasks can be formulated as trajectory tracking problems. The proposed multi-robot, multi-task transfer framework achieves high-accuracy trajectory tracking from the first iteration for a wide range of robot dynamics and desired trajectories.

In the realm of multi-robot transfer, the work in [4] learns an optimal static gain between the outputs of two linear, time-invariant (LTI), single-input, single-output (SISO) systems when they try to follow the same reference. The transformation error is minimized when the two systems have stable poles that lie close to each other. In [5], a preliminary study of transfer learning between two nonlinear, unicycle robots is presented. In [6], it is proved that the optimal transfer learning map between two robots is, in general, a dynamic system. An algorithm is also provided for determining the properties of the optimal dynamic map, including its order, relative degree and the variables it depends on. In [7], a data transfer mechanism based on manifold alignment of input-output data is proposed. The transferred data improves learning of a model of a robotic arm by using data from a different robotic arm. In [8], an algorithm for robust transfer of learned policies to target environments with unmodeled disturbances or unknown parameters is developed. This method improved performance on unmodeled disturbances by 50-300%, compared to naive transfer. However, this algorithm was not evaluated on physical systems.

Multi-task transfer is important to consider because learning approaches, such as iterative learning control (ILC), are usually not able to transfer knowledge from previously learned tasks to new, unseen tasks. In [9], using knowledge from previously learned trajectories, a linear map is created to calculate the inputs required to track unseen trajectories. Experimental results show that only one learned trajectory is needed to improve the performance on a new trajectory. However, the variables required for the linear map are selected through experimental trial-and-error, which may be time consuming. In [10], reinforcement learning is used to learn a value function for a particular load. Once learned, it is used to generate swing-free trajectories for aerial vehicles with suspended loads. Then, sufficient criteria are found to allow transfer of the learned policy to different models, states and action spaces. In [11] and [12], a deep neural network (DNN) is trained to achieve a unity map between the desired and actual outputs. The DNN adapts the reference signal to a feedback control loop to enhance the tracking performance of unseen trajectories. In [13], neural networks allow generalization of a task based on a single instance of the given task. However, the architecture of the neural network must be tailored to the specific task.

In the realm of multi-task and multi-robot transfer, the work in [14] trains neural network policies that can be decomposed into "task-specific" and "robot-specific" modules. When a new robot-task combination is encountered, the appropriate robot and task modules are composed to solve the problem. This architecture enables zero-shot generalization with a variety of robots and tasks in simulation. Neural network approaches require a significant amount of data and computational resources to train. In this paper, we emphasize data efficiency to achieve successful transfer in experiments.

The goal of this work is to design a learning architecture that is able to achieve high-accuracy tracking in the first iteration *(i)* despite the presence of changing dynamics which include switching the robot hardware altogether, and *(ii)* by using previously learned trajectories and generalizing knowledge to new, unseen trajectories. We propose a multi-task transfer framework placed on top of a multi-robot transfer framework (see Fig. 1).

The proposed, multi-robot transfer framework is based on the combined $\mathcal{L}_1$ adaptive control and ILC approach introduced in [15]. The ILC improves tracking performance over iterations, while the $\mathcal{L}_1$ adaptive controller forces dynamically-different nonlinear systems to behave as a specified linear model. Hence, trajectories learned on a system can be transferred among dynamically different systems (equipped with the same underlying $\mathcal{L}_1$ adaptive controller) to achieve high-accuracy tracking. The proposed, multi-task transfer scheme does not require exact knowledge of the system model or significant amounts of data as in [14]. Instead, it learns a map between a single desired trajectory and the inputs that make the system track the desired trajectory accurately. This map is used to calculate the inputs needed to track a new, unseen trajectory with high accuracy. Unlike [9], where trial-and-error is used, we derive the structure of this map based on concepts from control theory, the vector relative degree. The proposed approach also allows the system to continue learning over iterations after transfer.

The remainder of this paper is organized as follows. The problem is defined in Section III. The details of the proposed approach and the main results are presented in Section IV. Experimental results on two quadrotors are presented in Section V. Conclusions are provided in Section VI.

II. BACKGROUND

In this section, we review the definitions needed for the remainder of the paper. Consider a discrete-time, LTI, multi-input, multi-output (MIMO) system:

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

where $x(k) \in \mathbb{R}^n$ is the system state vector, $u(k) = [u_1(k), \ldots, u_p(k)]^T \in \mathbb{R}^p$ is the system input, $y(k) = [y_1(k), \ldots, y_p(k)]^T \in \mathbb{R}^p$ is the system output, k is the discrete-time index, and B and C are full rank. Let

$$
B = [B_1, \ldots, B_p] \quad \text{and} \quad C = [C_1^T, \ldots, C_p^T]^T.
$$

Definition 1 (Vector relative degree). System (1) is said to have a vector relative degree $(r_1, \ldots, r_p)$ if
 (i) $C_i A^k B_j = 0$, $\forall i = \{1, \ldots, p\}$, $\forall j = \{1, \ldots, p\}$ and $\forall k = \{0, \ldots, r_{i-2}\}$, and
 (ii) the decoupling matrix A_0 where $[A_0]_{ij} = C_i A^{r_i - 1} B_j$, $\forall i, j \in \{1, \ldots, p\}$ is nonsingular.

Remark 1. *If the system* (1) *has a vector relative degree* $(r_1, \ldots, r_p)$*, then we have*

$$
\begin{bmatrix}
y_1(k+r_1) \\
\vdots \\
y_p(k+r_p)
\end{bmatrix}
=
\begin{bmatrix}
C_1 A^{r_1} \\
\vdots \\
C_p A^{r_p}
\end{bmatrix}
x(k) + A_0
\begin{bmatrix}
u_1(k) \\
\vdots \\
u_p(k)
\end{bmatrix}.
\tag{2}
$$

For SISO systems, the relative degree is the number of sample delays between changing the input and observing a change of the output, and can be easily determined experimentally from the system's step response.

Remark 2. *Analogous to the case of SISO systems, the vector relative degree of a MIMO system can be determined from easy-to-execute experiments. In particular, one can carry out* p *experiments; in each experiment, one should apply a unit step input to only one of the* p *inputs, and monitor the responses of the* p *outputs. From Definition 1, a good estimate of the relative degree* r_i *is the number of sample delays between changing* any *of the inputs and seeing a change in the output* y_i*.

In other words, for the p easy-to-execute experiments, one can observe the minimum time delay obtained from the different experiments for each output dimension. After estimating the vector relative degree $(r_1, \ldots, r_p)$, one still

needs to verify that A_0 is invertible, see Definition 1. Note that from Remark 1, the rank of A_0 can be indirectly checked by studying the rank of Y_r where $[Y_r]_{ij} = y_i(r_i)|_j$ and $y_i(r_i)|_j$ is the value of the output y_i at time index r_i for the step input experiment j, and assuming that for the p experiments, the step inputs are applied at time index 0 and the system is initiated from the same initial condition $x_0 = 0$.

We next review the definition of zero dynamics. The *zero dynamics* of (1) are the invariant dynamics under which the system (1) evolves when the output y is constrained to be zero for all times. The zero dynamics represent the internal dynamics of the system when the system output is stabilized to zero. By representing the system (1) in the so-called Byrnes-Isidori normal form, it can be shown that the order of the minimum realization of the zero dynamics is $n-r$, where $r = \sum_{i=1}^p r_i$ [16], [17]. The system (1) is *minimum phase* if its zero dynamics are asymptotically stable in the Lyapunov sense. Based on the definition of zero dynamics and from (2), it can be shown that if the system (1) is minimum phase, then the input $u(k) = -A_0^{-1}[(C_1 A^{r_1})^T, \ldots, (C_p A^{r_p})^T]^T x(k)$ achieves asymptotic stability of the internal dynamics of (1). We use this fact in Section IV-B.

Finally, we review the definition of the projection operator, which will be used in Section IV-A. We define ∇ as the vector differential operator.

Definition 2 (Projection operator)**.** Consider a convex compact set with a smooth boundary given by $\Omega_c := \{\lambda \in \mathbb{R}^n | f(\lambda) \le c\}$, $0 \le c \le 1$, where $f : \mathbb{R}^n \to \mathbb{R}$ is the following smooth convex function:

$$f(\lambda) := \frac{(\epsilon_\lambda + 1)\lambda^T \lambda - \lambda_{max}^2}{\epsilon_\lambda \lambda_{max}^2}$$

with λ_{max} being the norm bound imposed on the vector λ, and $\epsilon_\lambda > 0$ is a projection tolerance bound of our choice. The projection operator is defined as [18]:

$$\text{Proj}(\lambda, y) := \begin{cases} y & \text{if } f(\lambda) < 0 \\ y & \text{if } f(\lambda) \ge 0 \text{ and } \nabla f^T y \le 0 \\ y - f_\lambda & \text{if } f(\lambda) \ge 0 \text{ and } \nabla f^T y > 0 \end{cases}$$

where $y \in \mathbb{R}^n$, $f_\lambda = \frac{\nabla f \nabla f^T y}{\|\nabla f\|^2} f(\lambda)$.

III. PROBLEM STATEMENT

The objective of this work is to achieve high-accuracy trajectory tracking in the first iteration in a multi-robot, multi-task framework, in which *(i)* there are uncertain, possibly changing environmental conditions, *(ii)* the training and testing robots are dynamically different, and *(iii)* the training and testing trajectories are different. We consider a control architecture as shown in Fig. 1. A multi-task framework is designed on top of a multi-robot framework.

The multi-robot framework combines an adaptive controller with a learning-based controller as in [15]. The adaptive controller forces two dynamically-different, minimum phase systems, which can be nonlinear systems, to behave in the same predefined way, as specified by a, possibly linear, reference model. The learning-based controller improves the tracking performance over iterations. Since dynamically different systems are forced to behave in the same predefined way, trajectories learned on a training system can be directly applied to a different testing system achieving high-accuracy trajectory tracking in the first iteration.

In the multi-task framework, information of the desired trajectory and the learned input trajectory $(\mathbf{y}_2^{*,l}, \mathbf{u}_2^l)$ for a single, previously learned trajectory l is given, where the subscript 2 is used to agree with the multi-robot framework notation, see Fig. 2. We aim to determine a map $\mathcal{M}$ from the desired outputs $\mathbf{y}_2^{*,l}$ to the learned inputs $\mathbf{u}_2^l$, such that the error between $\mathcal{M}\mathbf{y}_2^{*,l}$ and $\mathbf{u}_2^l$ is minimized, by solving:

$$\min_{\mathcal{M}} \|\mathcal{M}\mathbf{y}_2^{*,l} - \mathbf{u}_2^l\|. \tag{3}$$

We show in this paper that for linearized or LTI systems the map $\mathcal{M}$ is time- and state-invariant, and consequently, when a new desired trajectory $(l + 1)$ is encountered, an input trajectory that minimizes the trajectory tracking error can be calculated by: $\mathbf{u}_2^{l+1} = \mathcal{M}\mathbf{y}_2^{*,l+1}$. Unlike [9] where the structure of the map $\mathcal{M}$ is obtained through experimental trial-and-error, we use results from control systems theory to provide design guidelines for calculating the map $\mathcal{M}$.

IV. METHODOLOGY

In this section, we describe in detail the multi-robot and multi-task transfer architectures for MIMO systems.

A. Multi-robot transfer

The multi-robot transfer framework is based on the idea that an adaptive controller is able to force two dynamically different systems to behave in a predefined way. In this work we focus on MIMO systems. Therefore, any **MIMO $\mathcal{L}_1$ adaptive controller** implementation, such as [19], [20], can be used. For convenience and completeness, we present the MIMO $\mathcal{L}_1$ adaptive controller that we implemented in our experiments in Section V. The extended $\mathcal{L}_1$ adaptive controller assumes that the uncertain and changing dynamics of the robotic system can be described by a MIMO system for output feedback:

$$y_1(s) = A(s)(u_{\mathcal{L}_1}(s) + d_{\mathcal{L}_1}(s)), \quad y_2(s) = \frac{1}{s}y_1(s), \tag{4}$$

where $y_1(s)$ and $y_2(s)$ are the Laplace transforms of the translational velocity $y_1(t) \in \mathbb{R}^p$ and the position $y_2(t) \in \mathbb{R}^p$, respectively, $A(s)$ is a transfer function matrix of strictly-proper *unknown* transfer functions that can be stabilized by a proportional-integral controller, $u_{\mathcal{L}_1}(s)$ is the Laplace transform of the input $u_{\mathcal{L}_1}(t) \in \mathbb{R}^p$, and $d_{\mathcal{L}_1}(s)$ is the Laplace transform of disturbance signals defined as: $d_{\mathcal{L}_1}(t) := f(t, y_1(t))$, where $f : \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}^p$ is an *unknown* map subject to the global Lipschitz continuity assumption with Lipschitz constant L (see Assumption 4.1.1 in [18]).

The $\mathcal{L}_1$ adaptive output feedback controller aims to design a control input $u_{\mathcal{L}_1}(t)$ such that the output $y_2(t) \in \mathbb{R}^p$ tracks a bounded piecewise continuous reference input $u_2(t) \in \mathbb{R}^p$. We aim to achieve a desired closed-loop behavior by nesting the output of the $\mathcal{L}_1$ adaptive controller with output $y_1(t) \in$
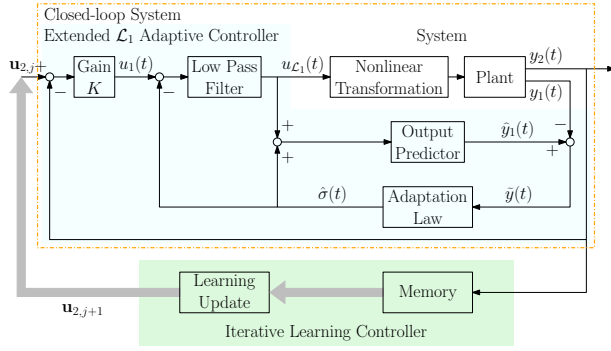
Fig. 2. The multi-robot transfer learning architecture combining the extended $\mathcal{L}_1$ adaptive control and iterative learning control. Figure adopted from [15]. The thick, light grey arrows represent signals of a whole trajectory execution.

$\mathbb{R}^p$ which tracks $u_1(t) \in \mathbb{R}^p$ within a proportional feedback loop (see Fig. 2). The desired inner loop behavior is given by the following first-order reference dynamic systems:

$$M(s) = \text{diag}(M_1(s), \ldots, M_p(s)), \ M_i(s) = \frac{m_i}{s + m_i}, \quad (5)$$

where $m_i > 0$. We can rewrite system (4) in terms of the reference system (5) as follows:

$$y_1(s) = M(s)(u_{\mathcal{L}_1}(s) + \sigma(s)), \quad y_2(s) = \frac{1}{s}y_1(s), \quad (6)$$

where the uncertainties in $A(s)$ and $d_{\mathcal{L}_1}(s)$ are combined into $\sigma(s)$, given by $\sigma(s) = M^{-1}(s)((A(s) - M(s))u_{\mathcal{L}_1}(s) + A(s)d_{\mathcal{L}_1}(s))$. The equations that describe the implementation of the extended $\mathcal{L}_1$ output feedback controller are:

Output Predictor: $\widehat{y}_1(s) = M(s)(u_{\mathcal{L}_1}(s) + \widehat{\sigma}(s))$, where $\widehat{\sigma}(s)$ is the adaptive estimate of $\sigma(s)$.

Adaptation Law: $\dot{\widehat{\sigma}}(t) = \Gamma\text{Proj}(\widehat{\sigma}(t), -\widetilde{y}(t))$, with $\widehat{\sigma}(0) = 0$, where $\widetilde{y}(t) := \widehat{y}_1(t) - y_1(t)$ and Proj(. , .) is defined in Section II. The adaptation rate $\Gamma > 0$ is subject to the lower bound specified in [18] and is set very large for fast adaptation.

Control Law: $u_{\mathcal{L}_1}(s) = V(s)(u_1(s) - \hat{\sigma}(s))$, where $V(s)$ is a diagonal transfer function matrix of low-pass filters:

$$V(s) = \text{diag}(V_1(s), \ldots, V_p(s)), \quad V_i(s) = \frac{\omega_i}{s + \omega_i},$$

where $\omega_i > 0$. By filtering out the high frequencies in $\hat{\sigma}(s)$, high adaptation gains can be used without deteriorating the transient behavior of the system.

Closed-Loop Feedback:

$$u_1(s) = K(u_2(s) - y_2(s)), \quad (7)$$

where $K = \text{diag}(K_1, \ldots, K_p)$, and $K_i \in \mathbb{R}^+$ is a proportional gain; used to make $y_2(t)$ track $u_2(t)$.

We design $V(s)$, K, and $M(s)$ such that $H(s) = A(s)(I - V(s) + V(s)M^{-1}(s)A(s))^{-1}$ and $F(s) = (sI + H(s)V(s)K)^{-1}$ are stable, and the following $\mathcal{L}_1$-norm condition is satisfied $\|F(s)H(s)(1 - V(s))\|_{\mathcal{L}_1}L < 1$, where L is the global Lipschitz constant of the disturbance function f.

The extended $\mathcal{L}_1$ adaptive controller makes the system

(roughly) behave as a linear, MIMO system described by:

$$y_2(s) = \text{diag}(D_1(s), \ldots, D_p(s))u_2(s), \ \text{where}$$
$$D_i(s) = \frac{K_i m_i}{s^2 + m_i s + K_i m_i}. \quad (8)$$

The multi-task transfer framework, discussed in the next subsection, requires a desired trajectory and the correct input that makes the system track the desired trajectory. To construct this pair of desired trajectory and corresponding correct input, we use an **optimization-based ILC** [21] to modify the input and improve the tracking performance of the system, which now behaves close to (8), in a small number of iterations $1, \ldots, j$. The objective is to make $y_2(t)$ track a desired output trajectory $y_2^*(t)$, which is defined over a finite-time interval. We assume that there exist input, state and output trajectories $(u_2^*(t), x^*(t), y_2^*(t))$ that are feasible with respect to the true dynamics of the $\mathcal{L}_1$-controlled system under linear constraints on the system inputs and/or outputs. We also assume that the system stays close to the desired trajectory; hence, we only consider small deviations from the above trajectories $(\widetilde{u}_2(t), \widetilde{x}(t), \widetilde{y}_2(t))$. The small deviation signals are discretized since the input of computer-controlled systems is sampled and measurements are only available at fixed time intervals. We use Equation (12) in [22] to write the output and input trajectories in the lifted representation as $\widetilde{\mathbf{y}}_2 = [\widetilde{y}_2^T(1), \ldots, \widetilde{y}_2^T(N)]^T$, where $N < \infty$ is the number of discrete samples, $\widetilde{y}_2(k) = [\widetilde{y}_{2,1}(k), \ldots, \widetilde{y}_{2,p}(k)]^T \in \mathbb{R}^p$ and $\widetilde{\mathbf{u}}_2 = [\widetilde{u}_2^T(0), \ldots, \widetilde{u}_2^T(N-1)]^T$ where $\widetilde{u}_2(k) = [\widetilde{u}_{2,1}(k), \ldots, \widetilde{u}_{2,p}(k)]^T \in \mathbb{R}^p$.

Now consider the minimum phase, discrete-time, LTI, MIMO system:

$$\widetilde{x}(k+1) = A_{\mathcal{L}_1}\widetilde{x}(k) + B_{\mathcal{L}_1}\widetilde{u}_2(k)$$
$$\widetilde{y}_2(k) = C_{\mathcal{L}_1}\widetilde{x}(k), \quad (9)$$

where $A_{\mathcal{L}_1}$, $B_{\mathcal{L}_1}$ and $C_{\mathcal{L}_1}$ are the discrete-time matrices that describe (8). To capture the dynamics of a complete trial by a static mapping, we compute the lifted representation of (9) using Equations (13-14) in [22] to obtain:

$$\widetilde{\mathbf{y}}_{2,j} = \mathbf{F}_{\text{ILC}}\widetilde{\mathbf{u}}_{2,j} + \widetilde{\mathbf{d}}_\infty \quad (10)$$

where the subscript j represents the iteration number, $\mathbf{F}_{\text{ILC}}$ is a constant matrix derived from the discretized model (9) and $\widetilde{\mathbf{d}}_\infty$ represents a repetitive disturbance that is initially unknown, but is identified during the learning process. In many control applications, constraints must be placed on the process variables to ensure safe and smooth operations. The system may be subject to linear input or output constraints:

$$S_c\widetilde{\mathbf{y}}_2 \leq \widetilde{\mathbf{y}}_{2,max}, \quad Z_c\widetilde{\mathbf{u}}_2 \leq \widetilde{\mathbf{u}}_{2,max}, \quad (11)$$

where S_c and Z_c are matrices of appropriate size.

Following Equations (19-26) in [22], we implement an iteration-domain Kalman filter to obtain an estimate of the disturbance $\widetilde{\mathbf{d}}_\infty$, $\widehat{\mathbf{d}}_{j|j}$, based on measurements from iterations $1, \ldots, j$. Using (10) and the estimated disturbance, the

estimated output error $\widehat{\mathbf{y}}_{2,j+1|j}$ can be represented by:

$$\widehat{\mathbf{y}}_{2,j+1|j} = \widetilde{\mathbf{F}}_{\mathrm{ILC}} \widetilde{\mathbf{u}}_{2,j+1} + \widehat{\mathbf{d}}_{j|j} \,. \tag{12}$$

At the end of iteration j, an update step computes the next input sequence $\widetilde{\mathbf{u}}_{2,j+1}$ that minimizes the estimated output error $\widehat{\mathbf{y}}_{2,j+1|j}$ and the control effort $\widetilde{\mathbf{u}}_{2,j+1}$ based on the following cost function:

$$J(\widetilde{\mathbf{u}}_{2,j+1}) = \widehat{\mathbf{y}}_{2,j+1|j} \mathbf{Q} \widehat{\mathbf{y}}_{2,j+1|j} + \widetilde{\mathbf{u}}_{2,j+1} \mathbf{R} \widetilde{\mathbf{u}}_{2,j+1} \,,$$

subject to (11), where $\mathbf{Q}$ is a constant, positive semi-definite matrix and $\mathbf{R}$ is a constant, positive definite matrix. The resulting convex optimization problem can be solved efficiently with state-of-the-art optimization libraries. In this work, we use the IBM ILOG CPLEX Optimizer.

Remark 3. *If the source and target systems have underlying $\mathcal{L}_1$ adaptive controllers with different reference models, then it is still possible to implement the multi-robot framework by using the reference models to build a map from the source system to the target system [6]. Using this map, trajectories learned on the source system can be transferred to the target system, which has a different reference model.*

B. Multi-task transfer

To facilitate transfer learning between different desired trajectories, a mapping is learned from desired trajectories to the inputs that make the system track these trajectories. Our proposed multi-task transfer learning framework uses insights from control systems theory and knowledge of a single previously learned trajectory to calculate the reference input of a new, unseen trajectory to minimize the trajectory tracking error. We showed that the input-output behavior of the system under the extended $\mathcal{L}_1$ adaptive control is LTI, MIMO and minimum phase, see (8). We use (1) to describe the controlled system and assume it is minimum phase.

Lemma 1. *Consider a minimum phase, discrete-time, MIMO, LTI system* (1), *and a smooth desired trajectory* $\mathbf{y}_2^*$. *Then, there exists a control input sequence* $\mathbf{u}_2$ *that achieves perfect tracking of* $\mathbf{y}_2^*$. *Moreover, at time instant k, the control input $u_2(k)$ can be represented as a linear combination of the state $x(k)$ and the values $y_{2,1}^*(k+r_1), \ldots, y_{2,p}^*(k+r_p)$, where $y_{2,i}^*(j)$ is the value of the ith component of the desired output at time index j, and $(r_1, \ldots, r_p)$ is the vector relative degree of the system.*

Proof. From Definition 1 the matrix A_0 is nonsingular. Hence, we can define the following control law:

$$\begin{bmatrix} u_{2,1}(k) \\ \vdots \\ u_{2,p}(k) \end{bmatrix} = A_0^{-1} \left(\begin{bmatrix} -C_1 A^{r_1} \\ \vdots \\ -C_p A^{r_p} \end{bmatrix} x(k) + \begin{bmatrix} y_{2,1}^*(k+r_1) \\ \vdots \\ y_{2,p}^*(k+r_p) \end{bmatrix} \right). \tag{13}$$

By assumption, the desired trajectory $y_2^*(k)$, $k \geq 0$, is known. Hence, at time step k, $y_{2,1}^*(k+r_1), \ldots, y_{2,p}^*(k+r_p)$ are all known, and (13) can be realized.

Next, by plugging the control law (13) into (2), we obtain:

$$\begin{bmatrix} y_1(k+r_1) \\ \vdots \\ y_p(k+r_p) \end{bmatrix} = \begin{bmatrix} y_{2,1}^*(k+r_1) \\ \vdots \\ y_{2,p}^*(k+r_p) \end{bmatrix}. \tag{14}$$

Since the system (1) is minimum phase by assumption, then as discussed in Section II, the control input $u_2(k) = -A_0^{-1}[(C_1 A^{r_1})^T, \ldots, (C_p A^{r_p})^T]^T x(k)$ achieves asymptotic stability of the internal dynamics of (1). Since for linear systems asymptotic stability implies bounded-input, bounded-output (BIBO) stability, then it is evident that under the control law (13) and for bounded reference input y_2^*, the internal states of (1) are bounded.

Finally, by applying the control law (13) at each time step k, $k \geq 0$, we have from (14), $y_i(\widetilde{k}) = y_{2,i}^*(\widetilde{k})$, for each $\widetilde{k} \geq r_i$, i.e., the perfect tracking condition is achieved. From the control law (13), the input $u_2(k)$ is a linear combination of $x(k)$ and $y_{2,1}^*(k+r_1), \ldots, y_{2,p}^*(k+r_p)$, which completes the proof. $\square$

Suppose that we are given a smooth desired trajectory $\mathbf{y}_2^*$ and using a learning approach, such as ILC, we are able to obtain for this particular desired trajectory an input sequence vector $\mathbf{u}_2$ that achieves high-accuracy tracking performance. Similar to [9], we use the desired output and learned input trajectories $(\mathbf{y}_2^*, \mathbf{u}_2)$ to learn a transfer learning map $\mathcal{M}$ that minimizes the transfer learning error as in (3). Unlike [9], in which the map is constructed through experimental trial-and-error, we use Lemma 1 to construct the map. In particular, we know from Lemma 1 that to achieve perfect tracking, $u_{2,i}(k)$, $i = 1, \ldots, p$, should be a linear combination of $x(k)$ and $y_{2,1}^*(k+r_1), \ldots, y_{2,p}^*(k+r_p)$, where $(r_1, \ldots, r_p)$ is the vector relative degree of (1). We assume, for now, that the state $x(k)$ can be measured or estimated, and stored. Hence, we propose to build, with the available information, the following windowing function:

$$W(\mathbf{x}, \mathbf{y}_2^*) = \begin{bmatrix} x^T(0) & \bar{y}_2^*(0) \\ \vdots & \vdots \\ x^T(N_r) & \bar{y}_2^*(N_r) \end{bmatrix}, \tag{15}$$

where $\bar{y}_2^*(a) = [y_{2,1}^*(a+r_1), \ldots, y_{2,p}^*(a+r_p)]$, and $N_r = N - \max_{j \in \{1, \ldots, p\}}(r_j)$. Using the windowing function $W(\mathbf{x}, \mathbf{y}_2^*)$, we define the following learning process:

$$\mathbf{u}_{2,i} = W(\mathbf{x}, \mathbf{y}_2^*) \theta_i \,, \tag{16}$$

where $\mathbf{u}_{2,i} = [u_{2,i}(0), \ldots, u_{2,i}(N_r)]^T$ is the collection of the ith elements of $\mathbf{u}_2$, obtained from the ILC algorithm. This is a linear regression problem for the parameter vector $\theta_i \in \mathbb{R}^{n+p}$.

Remark 4. *The vectors of unknowns θ_i, $i \in \{1, \ldots, p\}$, are all functions of the system matrices A, B, C, which is evident from (13) and the definition of the matrix A_0 in*

Definition 1. We emphasize that the vectors θ_i do not depend on the desired trajectory $\mathbf{y}_2^*$ or the system states.

Therefore, we can reuse the calculated vectors θ_i, which build an invariant map, to calculate for new, unseen, desired trajectories correct input vectors that achieve perfect tracking. In particular, we use the vectors θ_i to calculate the control input that achieves perfect tracking of a new desired trajectory $\mathbf{y}_2^{*,new}$ as follows:

$$u_{2,i}^{new}(k) = \begin{bmatrix} x^T(k) & \bar{y}_2^{*,new}(k) \end{bmatrix} \theta_i \quad \forall i \in \{1, \ldots, p\}, \tag{17}$$

where θ_i, $i \in \{1, \ldots, p\}$ are calculated by (16) and $\bar{y}_2^{*,new}(k) = [y_{2,1}^{*,new}(k + r_1), \ldots, y_{2,p}^{*,new}(k + r_p)]$.

Remark 5. *Our proposed control law (17) does not assume the knowledge of the system matrices A, B and C as in (13). It only assumes the knowledge of the vector relative degree of the system, which can be calculated from (8), or obtained through experiments, see Remark 2.*

Notice that the construction of the windowing function $W(\mathbf{x}, \mathbf{y}_2^*)$ requires the knowledge of the system states or estimated values of the states. We now discuss how the proposed transfer learning approach can be extended to the case where the state measurements or their estimated values are not available. To this end, we review an important Lemma from [23].

Lemma 2. *Consider the discrete-time, LTI system (1), and suppose that the pair (A, C) is observable. Then, the system state $x(k)$ is given uniquely in terms of input-output sequences as follows:*

$$x(k) = M_u \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-\bar{N}) \end{bmatrix} + M_y \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-\bar{N}) \end{bmatrix}, \tag{18}$$

where $\bar{N}$ is an upper bound on the system's observability index, $M_u = U_{\bar{N}} - M_y T_{\bar{N}}$, $M_y = A^{\bar{N}} V_{\bar{N}}^+$, $V_{\bar{N}}^+ = (V_{\bar{N}}^T V_{\bar{N}})^{-1} V_{\bar{N}}^T$ is the left inverse of the observability matrix $V_{\bar{N}}$, and

$$U_{\bar{N}} = \begin{bmatrix} B & AB & \ldots & A^{\bar{N}-1}B \end{bmatrix},$$

$$T_{\bar{N}} = \begin{bmatrix} 0 & CB & CAB & \ldots & CA^{\bar{N}-2}B \\ 0 & 0 & CB & \ldots & CA^{\bar{N}-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & & 0 & CB \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$V_{\bar{N}} = \begin{bmatrix} (CA^{\bar{N}-1})^T & \ldots & (CA)^T & C^T \end{bmatrix}^T.$$

In other words, if the system (1) is observable, then the state vector can be represented as a linear combination of a

finite sequence of past inputs and outputs of the system. From Lemmas 1 and 2, it can be shown that the control input $u_2(k)$ that achieves perfect tracking of the desired trajectory can be represented as a linear combination of $u_2(k-1), \ldots, u_2(k-\bar{N}), y(k-1), \ldots, y(k-\bar{N}), y_{2,1}^*(k+r_1), \ldots, y_{r,p}^*(k+r_p)$, where $y(k)$ is the actual output of the system at time step k, $(r_1, \ldots, r_p)$ is the vector relative degree of (1) and $\bar{N}$ is an upper bound of the observability index of (1).

Hence, for observable systems, our approach is still valid when only input-output data is available. In this case, the ILC algorithm will also be used to calculate an input $\mathbf{u}$ that will make the output $\mathbf{y}$ track the desired trajectory $\mathbf{y}_2^*$ with high accuracy. Using this input-output information, we can redefine the windowing function (15) by substituting the state $x(k)$ with a finite sequence of input-output information such that:

$$W_{IO}(\mathbf{u}, \mathbf{y}, \mathbf{y}_2^*) = \begin{bmatrix} \bar{u}(\bar{N}) & \bar{y}(\bar{N}) & \bar{y}_2^*(\bar{N}) \\ \vdots & \vdots & \vdots \\ \bar{u}(N_r) & \bar{y}(N_r) & \bar{y}_2^*(N_r) \end{bmatrix}, \tag{19}$$

where $\bar{y}_2^*(a) = [y_{2,1}^*(a + r_1), \ldots, y_{2,p}^*(a + r_p)]$, $\bar{u}(a) = [u^T(a - 1), \ldots, u^T(a - \bar{N})]$, and $\bar{y}(a) = [y^T(a - 1), \ldots, y^T(a - \bar{N})]$. Similar to (16), we then calculate the vectors of unknown parameters $\theta_{IO,i}$. The vectors are functions of the matrices of system (1) and can be used to calculate the required input when a new, unseen trajectory is encountered. For this case, the proposed control law is:

$$u_{2,i}^{new}(k) = \begin{bmatrix} \bar{u}(k) & \bar{y}(k) & \bar{y}_2^{*,new}(k) \end{bmatrix} \theta_{IO,i}, \; \forall i \in \{1, \ldots, p\}.$$

In order to allow the system to continue learning after transfer, we need to provide the ILC with an initial estimate of the repetitive disturbance $\widehat{\mathbf{d}}_{0,trans}$. We assume that the calculated input $\mathbf{u}_2^{new}$ achieves perfect tracking of the new trajectory; hence, $\widehat{\mathbf{y}}_{2,j+1|j}$ in (12) is 0. Using (12) and the calculated input $\mathbf{u}_2^{new}$, we are able to calculate the disturbance $\widehat{\mathbf{d}}_{0,trans}$.

In this subsection, we derived the *multi-task* transfer learning framework for linear/linearized systems. This is not a big restriction given that the underlying $\mathcal{L}_1$ adaptive controller is able to force a nonlinear system to approximately behave as a linear model. In fact, the $\mathcal{L}_1$ adaptive controller can be replaced by another adaptive controller with a reference model, e.g., a model reference adaptive controller. However, we implement the $\mathcal{L}_1$ adaptive controller as it provides performance bounds on the output and input signals [18]. Nevertheless, it should be noted that the proposed framework can be also extended to **nonlinear systems** with well-defined vector relative degrees and stable inverse dynamics. In particular, analogous to Lemma 1, it can be shown that there exists a control input satisfying perfect tracking of an arbitrary, smooth trajectory $\mathbf{y}_2^*$, and this input is a nonlinear function of the state $x(k)$ and the values $y_{2,1}^*(k+r_1), \cdots, y_{2,p}^*(k+r_p)$, where $(r_1, \ldots, r_p)$ is the system's vector relative degree. In practice, one can use a nonlinear regression model, e.g.,

Fig. 3. The two different quadrotors used in the experiments. The quadrotor on the left is the Bebop 2, and on the right is the AR.Drone 2.0.

a neural network or a Gaussian Process, to approximate this unknown nonlinear function [11], [12]. Alternatively, smooth nonlinear systems can be approximated by piecewise affine/linear systems with arbitrary accuracy [24]. One can construct a cover/partition of the state space of a nonlinear system, and represent the nonlinear system in each region of the cover/partition with a local, affine/linear model. One can use the above proposed results to define a linear transfer learning map for each local region, resulting overall in a piecewise linear transfer learning map. The details will be further studied in future publications.

V. EXPERIMENTAL RESULTS

This section shows the experimental results of the proposed multi-robot, multi-task framework applied to quadrotors for high-accuracy trajectory tracking. We assess three aspects to verify the effectiveness of the proposed framework: *(i)* capability to transfer different trajectories between different quadrotor platforms, *(ii)* repeatability of results, and *(iii)* use of different reference models for the $\mathcal{L}_1$ adaptive controllers of the source and target systems. These experiments significantly extend the experiments in [15], which only assessed the multi-robot transfer learning framework.

The vehicles used in the experiments are the Parrot AR.Drone 2.0 and the Parrot Bebop 2 (see Fig. 3). Each quadrotor has an underlying $\mathcal{L}_1$ adaptive controller that makes both vehicles behave close to a reference system. In what follows, the signals $u_{1,i}(t)$ are the desired translational velocities, $u_{2,i}(t)$ are the desired translational positions, $y_{1,i}(t)$ are the quadrotor translational velocities and $y_{2,i}(t)$ are the quadrotor translational positions in the $i = x, y, z$ directions, respectively. A central overhead motion capture camera system provides position, roll-pitch-yaw Euler angles and rotational velocity measurements, and through numerical differentiation, we obtain translational velocities. We propose six different trajectories to test our approach, as shown in Fig. 4. To quantify the tracking performance, an average position error along the trajectory is defined by:

$$e = \frac{1}{N} \sum_{i=1}^{N} \sqrt{e_x^2(i) + e_y^2(i) + e_z^2(i)}, \qquad (20)$$

where $e_j(i) = u_{2,j}(i) - y_{2,j}(i)$ and $j = x, y, z$.

A. Multi-robot, multi-task transfer: capabilities

To assess the performance of the proposed multi-robot, multi-task transfer framework under different conditions, we generate six different trajectories (see Fig. 4) to be transferred
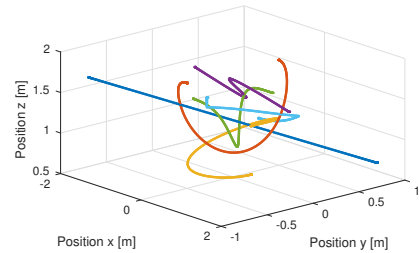

Fig. 4. The six different trajectories that are used to test the multi-robot, multi-task transfer framework. The source and target trajectories used to assess repeatability and different reference models are depicted in dark blue and orange, respectively.
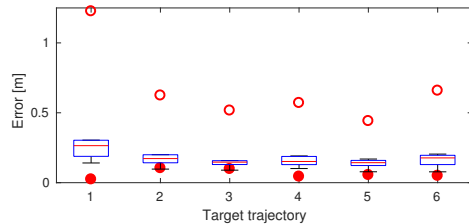

Fig. 5. Tracking errors of six learning processes without transfer information on the Bebop 2 at iteration 1 (open circles) and iteration 10 (filled circles). Boxplots summarize the six one-to-all experiments of the multi-robot, multi-task framework, which significantly decreases the error in the first iteration after transfer. On each box, the red mark indicates the median, while the bottom and top edges indicate the 25th and 75th percentiles, respectively. The whiskers represent the most extreme data points.

from the AR.Drone 2.0 to the Bebop 2. We learn each of the six trajectories on the AR.Drone 2.0 using the multi-robot framework (Section IV-A). We devise a *one-to-all* transfer scheme, in which we use one of the six trajectories as a source trajectory to transfer to each of the six trajectories, now called target trajectories (including the source trajectory, which is the the multi-robot transfer learning case). Using the multi-task framework (Section IV-B), we apply the one-to-all scheme six times such that each of the six trajectories is the source trajectory once (6 source × 6 target trajectories). Fig. 5 shows, for each of the six target trajectories, the tracking error on a Bebop 2 during a learning process at iteration 1 (open circles) and at iteration 10 (filled circles) when no transfer information is used. The tracking errors in the first iteration after applying the one-to-all, multi-robot, multi-task transfer are summarized in six boxplots, each of which corresponds to one target trajectory. The red mark on each box indicates the median, while the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers represent the most extreme data points. Using the proposed framework significantly reduces the error in the first iteration. On average, the percentage of error reduction for the 36 experiments is 74.12%, when using *only 6 seconds* of training data (length of each trajectory). This is a significant improvement compared to *(i)* [9] where the structure of the transfer map is determined by trial-and-error, and *(ii)* [14] where the training of neural networks requires a significant amount of data.

B. Multi-robot, multi-task transfer: repeatability

We also test the repeatability of the above results. We choose a single pair of source and target trajectories (in
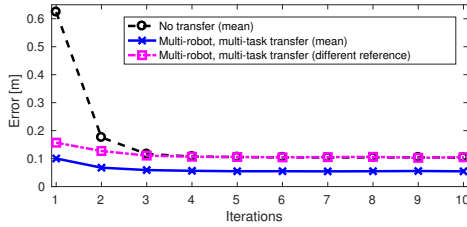
Fig. 6. For a single source and target trajectory pair, the average position errors over ten 10-iteration ILC processes when the Bebop 2 is tracking the target trajectory are shown *(i)* in blue, when the proposed multi-robot (AR.Drone 2.0 to Bebop 2), multi-task (source to target trajectory) framework is used, and *(ii)* in black, when no transfer information is used. The errors after the proposed multi-robot, multi-task transfer when the source and target systems have $\mathcal{L}_1$ adaptive controllers with different reference models are shown in magenta.

Fig. 4, dark blue and orange trajectory, respectively). We repeat ten times a 10-iteration learning process on the Bebop 2 with underlying $\mathcal{L}_1$ adaptive controller and ILC when *(i)* no transfer is used, and *(ii)* transfer information from the source trajectory learned on the AR.Drone 2.0 is used to initialize the learning process. Fig. 6 shows the mean error over the ten repetitions for each of the 10 iterations. The proposed framework significantly decreases the tracking error in the first iteration. The standard deviation is very low, with a mean of 0.0013 m and 0.0016 m for the no transfer and transfer cases, respectively. This is achieved as the $\mathcal{L}_1$ adaptive controller forces the Bebop 2 to behave like the reference model despite the presence of disturbances.

C. Multi-robot, multi-task transfer: different reference models for the $\mathcal{L}_1$ adaptive controllers

For completeness, we include experimental results for Remark 3, when the source and target systems have $\mathcal{L}_1$ adaptive controllers with different reference models. In particular, we modified the reference model of the Bebop 2, and used a mapping between the reference models in addition to the proposed multi-robot, multi-task transfer framework. The tracking error of a 10-iteration ILC process using transfer information is shown in magenta in Fig. 6. The proposed framework reduces the tracking error in the first iteration after transfer by 74.86%, even when the $\mathcal{L}_1$ adaptive controller of the target system has a different reference model.

VI. CONCLUSIONS

In this paper, we introduced a multi-robot, multi-task transfer learning framework for MIMO systems. We focused on the trajectory tracking problem. The multi-robot transfer learning framework is based on a combined $\mathcal{L}_1$ adaptive controller and ILC. The $\mathcal{L}_1$ adaptive controller forces systems to behave like a specified linear reference model, allowing learned tasks to be directly transferred to other systems. The multi-task transfer learning framework uses control theory results to build a time- and state-invariant map from the desired trajectory to the input that accurately tracks this trajectory. This map can be used to generate inputs for new desired trajectories. Experimental results on two different quadrotors and six different trajectories show that the proposed framework reduces the first-iteration tracking

error by 74% on average, when information from tracking a different, single trajectory on a different quadrotor is utilized.

REFERENCES

[1] R. Skelton, "Model error concepts in control design," *International Journal of Control*, vol. 49, no. 5, pp. 1725–1753, 1989.

[2] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[3] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*. Wiley New York, 2007, vol. 2.

[4] K. V. Raimalwala, B. A. Francis, and A. P. Schoellig, "An upper bound on the error of alignment-based transfer learning between two linear, time-invariant, scalar systems," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5253–5258.

[5] ——, "A preliminary study of transfer learning between unicycle robots," in *AAAI Spring Symposium Series*, 2016.

[6] M. K. Helwa and A. P. Schoellig, "Multi-robot transfer learning: A dynamical system perspective," *arXiv preprint arXiv:1707.08689*, Accepted at IROS 2017.

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

[8] J. Harrison, A. Garg, B. Ivanovic, Y. Zhu, S. Savarese, L. Fei-Fei, and M. Pavone, "Adapt: zero-shot adaptive policy transfer for stochastic dynamical systems," *arXiv preprint arXiv:1707.04674*, 2017.

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

[10] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Learning swing-free trajectories for UAVs with a suspended load," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 4902–4909.

[11] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," *arXiv preprint arXiv:1705.10932*, Accepted at CDC 2017.

[12] 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 International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5183–5189.

[13] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *arXiv preprint arXiv:1703.07326*, 2017.

[14] 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 International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2169–2176.

[15] K. Pereida, R. R. Duivenvoorden, and A. P. Schoellig, "High-precision trajectory tracking in changing environments through $\mathcal{L}_1$ adaptive feedback and iterative learning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 344–350.

[16] A. Isidori, *Nonlinear Control Systems*. Springer-Verlag London Limited, 1995.

[17] M. A. Henson and D. E. Seborg, "Feedback linearizing control," *Nonlinear process control*, pp. 149–231, 1997.

[18] N. Hovakimyan and C. Cao, $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2010.

[19] H. Jafamejadsani, H. Lee, N. Hovakimyan, and P. Voulgaris, "A multirate adaptive control for mimo systems with application to cyber-physical security," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2017 [Submitted].

[20] H. Lee, "$\mathcal{L}_1$ adaptive control for nonlinear and non-square multivariable systems," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2017 [Submitted to IEEE TAC].

[21] A. P. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proc. of the European Control Conference (ECC)*, 2009, pp. 1505–1510.

[22] A. P. Schoellig, F. L. Mueller, and R. D'Andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, pp. 103–127, 2012.

[23] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 14–25, 2011.

[24] M. K. Helwa and P. E. Caines, "Epsilon controllability of nonlinear systems on polytopes," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2015, pp. 252–257.