2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Knowledge Transfer for High-Performance Quadrocopter Maneuvers

Michael Hamer, Markus Waibel and Raffaello D'Andrea

Abstract— **Iterative Learning Control algorithms are based on the premise that "practice makes perfect". By iteratively performing an action, repetitive errors can be learned and accounted for in subsequent iterations, in a non-causal and feed-forward manner. This method has been previously implemented for a quadrocopter system, enabling the quadrocopter to learn to accurately track high-performance slalom trajectories. However, one major limitation of this system is that knowledge from previously learned trajectories is not generalized or transferred to new trajectories; these must be learned from a state of zero experience. This paper experimentally shows that the major dynamics of the Iterative Learning Control process can be captured by a linear map, trained on previously learned slalom trajectories. This map enables this prior knowledge to be used to improve the initialization of an unseen trajectory. Experimental results show that prediction based on a single prior is enough to reduce the initial tracking error for an unseen trajectory by an order of magnitude.**

I. Introduction

Pioneered in [1], Iterative Learning Control (ILC) algorithms are one method for improving the tracking accuracy of repeated trajectories. These algorithms monitor tracking errors along a repeated trajectory, and compensate for these errors by modifying the input to the next iteration. An overview of ILC algorithms and results is presented in two recent surveys on the topic and the references therein [2], [3]. These results demonstrate the effectiveness of ILC algorithms at compensating for repetitive disturbances along a single trajectory.

This paper focuses on a subclass of ILC algorithms, namely Indirect Iterative Learning Control (IILC), summarized in [3]. While ILC directly alters the open-loop input to the system, IILC alters the input trajectory given to a stabilizing feedback controller, which then converts this input trajectory into system control inputs. For the purposes of this paper, it is assumed that the input trajectory has the same units as the desired output trajectory; for example, position is both commanded and desired.

Although ILC (and by extension, IILC) algorithms are effective at improving tracking error for a single trajectory, the results of trajectory learning are not generalized, and thus new trajectories must be learned from a state of zero experience (ie. assuming zero disturbance and nominal model dynamics). This problem has been previously addressed in [4]–[8].

The problem of knowledge transfer in ILC is addressed in [4], where the authors propose using locally weighted learning to build a database of previous trajectory segments, and successfully use a nearest-neighbor approach to find similar historical trajectory segments to improve the ILC initialization for an unknown trajectory.

A second solution to the problem of ILC knowledge transfer is presented in [5], [6], where the authors model the system's inverse dynamics using a neural network. This approach was later adopted in [9].

A third approach, presented in [7], is to apply approximate fuzzy logic methods to the previously learned trajectories, and to use these methods to construct improved predictions.

Presented in [8] is a fourth approach to the problem of ILC knowledge transfer, where the authors use a matrix transformation to capture the difference between a previously executed trajectory and an unseen trajectory. This transformation and knowledge of the previously learned input is then used to predict the required input for the unseen trajectory.

All four of the previously discussed approaches have been successfully demonstrated on various physical systems.

This paper takes a different approach to the problem of knowledge transfer, focusing specifically on IILC. Experimental results show that the major dynamics of the IILC learning process itself (as opposed to the acted-upon system) can be approximated by a linear mapping around the class of desired trajectories. This approach, which we named "Windowed Learning Transfer" (WLT) for the purposes of this paper, is presented in detail in Section II.

The WLT method considers the required input at each point in time as a linear combination of previous and future states from the nominal trajectory. These previous and future states are compiled into a time-window, to which weighting factors are fitted using a linear least-squares approach. Due to its simple, least-squares-based structure, the proposed method is extremely fast and requires no complex training algorithms.

The effectiveness of WLT at reducing initial tracking error is experimentally demonstrated in Section III, where we show an application to a quadrocopter slalom learning experiment [10]. In this experiment, an optimization-based IILC algorithm enables a quadrocopter to learn to fly a high-performance trajectory through an arbitrary configuration of slalom poles. Experimental results show that once a single slalom course has been flown, the reference input for a subsequent, arbitrary slalom configuration can be predicted using WLT, and the initial tracking error can be reduced from an average RMS error of 1.47m when initialized nominally, to 0.20m when initialized with a WLT prediction trained on a single prior. The WLT prediction often allows the slalom

The authors are with the Institute for Dynamic Systems and Control, ETH Zürich, Switzerland {hamerm,mwaibel,rdandrea}@ethz.ch

course to be successfully flown without additional learning[1].

II. Preliminaries

In this section, IILC and WLT are defined mathematically. Assumptions made by this mathematical construction are also highlighted.

A. Indirect Iterative Learning Control

For the purposes of this paper, we define the IILC problem as follows. Consider a feedback-controlled system $\mathcal{F}$, which when given a time-lifted[2] input trajectory u, produces a time-lifted output trajectory s. This system is subject to stochastic disturbances, modeled as the zero-mean random variable d, and is mathematically represented as:

$$s = \mathcal{F}(u, d). \tag{1}$$

Consider now the problem of tracking the k^{th} desired trajectory. This desired trajectory is represented in the time-lifted domain as $\bar{s}^k$. IILC addresses the problem of tracking $\bar{s}^k$ by iteratively improving the input to the system based on the output tracking error from previous trials.

Using subscript to denote the iteration-index, the IILC process is initialized with $u_0^k = \hat{u}^k$, where $\hat{u}^k$ is an estimation of the input required to achieve $\bar{s}^k$ and is often calculated from a nominal understanding of $\mathcal{F}$.

Applying the input of the j^{th} iteration u_j^k to the system in (1), produces the output trajectory s_j^k. Through comparison of s_j^k with the desired trajectory $\bar{s}^k$, the tracking error at each point along the trajectory is computed and used to improve u_{j+1}^k, the input applied during the next iteration [11].

As the IILC process iteratively improves the input, the system's output converges to the desired output trajectory [11].

After m iterations, the results of the IILC process satisfy an application-defined stopping criteria. At this point, the input u_m^k is considered to be the result of the IILC process.

From this point forward, only the final result of the IILC learning process is considered. For this reason we drop the iteration subscript and rewrite the result as u^k.

We then summarize the IILC process described above as:

$$u^k = \mathbf{IILC}_{\mathcal{F}}(\, \bar{s}^k \,,\, \hat{u}^k \,). \tag{2}$$

B. Problem Representation

As per (2), the IILC process is initialized with $\hat{u}^k$, a prediction of the input trajectory required to achieve the desired trajectory $\bar{s}^k$. After the application-defined stopping criteria of the IILC process are satisfied, the satisfying input trajectory u^k is returned.

The WLT method presented in this paper is one method of extracting deterministic behavior from previous IILC results and using this knowledge to improve the initialization of future IILC processes. For each prior IILC result, the WLT method is trained on the desired trajectory $\bar{s}^k$ and the learned input trajectory u^k. Note that both trajectories can be trajectories in more than one spatial dimension. This paper focuses on the learning of trajectories in a flat plane, defined by x and y coordinates, thus $\bar{s}^k$ and u^k are trajectories in both x and y.

Given a collection of desired trajectories $(\bar{s}^0 \cdots \bar{s}^k)$ and the respective learned input trajectories $(u^0 \cdots u^k)$, the task is then to determine the mapping $\mathcal{M}$ such that given an unseen desired trajectory $\bar{s}^{k+1}$ the required input u^{k+1} can be predicted. We therefore assume the existence of a mapping $\mathcal{M}$ such that:

$$\bar{s}^i \xrightarrow{\mathcal{M}} u^i \quad \forall i \in [0, k]. \tag{3}$$

In addition to existence, we also assume that $\mathcal{M}$ is time- and state-invariant, remaining constant across all trajectories and time-windows.

Once the mapping $\mathcal{M}$ has been found, prediction of u^{k+1} given $\bar{s}^{k+1}$ follows as:

$$\bar{s}^{k+1} \xrightarrow{\mathcal{M}} \hat{u}^{k+1}. \tag{4}$$

This method of learning transfer (akin to those presented in [4]–[8]), serves to improve the initialization accuracy of the IILC system, rather than acting as a replacement. Prediction errors made in $\hat{u}^{k+1}$ will thus be accounted for by the IILC algorithm during subsequent iterations.

C. Constructing a time-window

Central to WLT is the idea of a time-window. We assume that every point of the required input trajectory u^k can be approximated by a linear combination of the surrounding points in the desired trajectory $\bar{s}^k$.

Data points towards the beginning and ends of the trajectory are accounted for by extending the trajectory, assuming that it begins and ends at standstill. Using superscript to denote trajectory-index and square-braces to denote discrete-time index, we define:

$$\gamma^k[j] = \begin{cases} \bar{s}^k[0] & \text{if } j < 0 \\ \bar{s}^k[j] & \text{if } 0 \leq j < N_k \\ \bar{s}^k[N_k - 1] & \text{if } j \geq N_k \end{cases}. \tag{5}$$

Based on the above, a windowing function operating on a time-lifted reference trajectory $\bar{s}^k$, is defined as:

$$\mathcal{W}(\bar{s}^k) = \begin{bmatrix} \gamma^k[0-H] & \cdots & \gamma^k[0] & \cdots & \gamma^k[0+H] \\ \gamma^k[1-H] & \cdots & \gamma^k[1] & \cdots & \gamma^k[1+H] \\ \vdots & & \vdots & & \vdots \\ \gamma^k[N-H] & \cdots & \gamma^k[N] & \cdots & \gamma^k[N+H] \end{bmatrix}. \tag{6}$$

The horizon parameter H defines how many future and past data points are included in the window. Based on this, the actual window width is given by $W = 2H + 1$. Using the definitions from (5), (6), we define the learning process,

[1] A video demonstration entitled "Quadrocopter Slalom: Learning from prior experience" is publicly available at http://www.flyingmachinearena.org/videos/

[2] A time-lifted trajectory is defined to be the stacked vector of all time points of the trajectory, ie. $u = \begin{bmatrix} u[0] & u[1] & \cdots & u[N] \end{bmatrix}^{\mathsf{T}}$

based on $k+1$ previous IILC results, as:

$$\begin{bmatrix} u^0 \\ u^1 \\ \vdots \\ u^k \end{bmatrix} = \begin{bmatrix} \mathcal{W}(\bar{s}^0) \\ \mathcal{W}(\bar{s}^1) \\ \vdots \\ \mathcal{W}(\bar{s}^k) \end{bmatrix} \cdot \theta. \tag{7}$$

This is a least-squares problem for the parameter vector θ, where the u^i and $\bar{s}^i$ represent complete trajectories in the time-lifted domain, not individual points on a trajectory. Once fitted, the parameter vector θ can be used for prediction:

$$\hat{u}^{k+1} = \mathcal{W}(\bar{s}^{k+1}) \cdot \theta, \tag{8}$$

thus addressing the problem statement, outlined in Subsection II-B.

III. Transfer of Prior Learning to Unseen Quadrocopter-Slalom Trajectories

In this section, the WLT methodology is developed with respect to a quadrocopter slalom flying experiment.

A. Experimental Setup

Shown in Fig. 1, the experimental setup used to generate IILC-learned trajectories employs IILC to enable a quadrocopter to learn and precisely fly a slalom trajectory between an arbitrary configuration of poles.

This experiment was first presented in [10], where the authors demonstrated that the desired trajectory for an arbitrary configuration of slalom poles can be accurately and repeatably learned using IILC. The IILC process is initialized with the configuration's desired trajectory and a nominal input trajectory, and uses the optimization-based IILC approach presented in [11] to adapt the input trajectory based on tracking errors from previous iterations. This process is continued until the desired trajectory is accurately followed.

In this experimental setup, the quadrocopter is controlled by a feedback controller, operating at 50Hz. This controller is provided with a reference trajectory in both x and y, and with constant z and yaw coordinates. The IILC process operates in an indirect manner by updating the x and y input to the system's controller based on previous position errors. The described system interconnection is shown in Fig. 2.

B. Windowed Learning Setup

As described in Subsection III-A, the slalom experiment learns trajectories in both x and y dimensions. With reference to (6)–(7), and using subscript x and y to denote the respective trajectory components, we define the learning problem, where $\theta_{\mathrm{xx}}, \theta_{\mathrm{xy}}, \theta_{\mathrm{yx}}, \theta_{\mathrm{yy}} \in \mathbb{R}^{(2H+1)\times 1}$, as:

$$\begin{bmatrix} u_{\mathrm{x}}^0 & u_{\mathrm{y}}^0 \\ u_{\mathrm{x}}^1 & u_{\mathrm{y}}^1 \\ \vdots & \vdots \\ u_{\mathrm{x}}^k & u_{\mathrm{y}}^k \end{bmatrix} = \begin{bmatrix} \mathcal{W}(s_{\mathrm{x}}^0) & \mathcal{W}(s_{\mathrm{y}}^0) \\ \mathcal{W}(s_{\mathrm{x}}^1) & \mathcal{W}(s_{\mathrm{y}}^1) \\ \vdots & \vdots \\ \mathcal{W}(s_{\mathrm{x}}^k) & \mathcal{W}(s_{\mathrm{y}}^k) \end{bmatrix} \cdot \begin{bmatrix} \theta_{\mathrm{xx}} & \theta_{\mathrm{yx}} \\ \theta_{\mathrm{xy}} & \theta_{\mathrm{yy}} \end{bmatrix}. \tag{9}$$

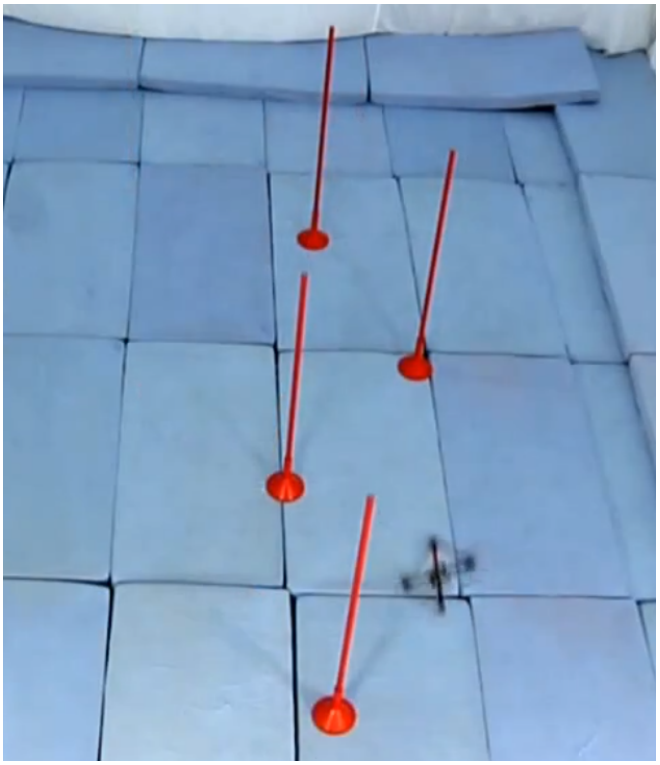

Fig. 1. A photograph of the quadrocopter slalom experiment, showing a quadrocopter flying an aggressive trajectory through a set of poles, which define the course. In this experiment, IILC is used to enable the quadrocopter to accurately follow this high-performance trajectory. The WLT method builds upon this setup, allowing the transfer of learning between different slalom configurations, thus improving the initial slalom performance when compared with a nominal initialization. Courtesy of [10].

For example:

$$u_{\mathrm{x}}^0 = \begin{bmatrix} \mathcal{W}(s_{\mathrm{x}}^0) & \mathcal{W}(s_{\mathrm{y}}^0) \end{bmatrix} \cdot \begin{bmatrix} \theta_{\mathrm{xx}} \\ \theta_{\mathrm{xy}} \end{bmatrix} \tag{10a}$$

$$u_{\mathrm{y}}^0 = \begin{bmatrix} \mathcal{W}(s_{\mathrm{x}}^0) & \mathcal{W}(s_{\mathrm{y}}^0) \end{bmatrix} \cdot \begin{bmatrix} \theta_{\mathrm{yx}} \\ \theta_{\mathrm{yy}} \end{bmatrix}. \tag{10b}$$

This formulation allows for cross coupling terms between x and y trajectories (given by parameter vectors θ_{xy} and θ_{yx}). Further extensions are possible, such as the introduction of a

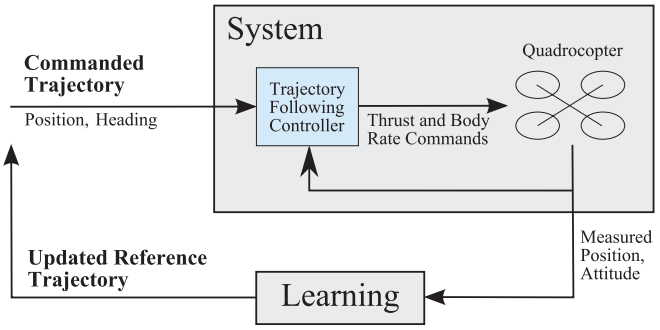

Fig. 2. The block-diagram interconnection of the system's components. The quadrocopter is controlled via a trajectory following feedback controller. This feedback controller receives an input from the ILC system, adjusted based on position tracking errors from previous attempts. Courtesy of [10].

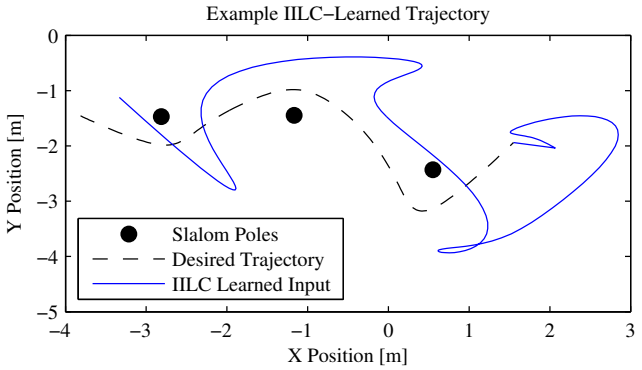

Fig. 3. The blue input trajectory is learned by the IILC system as an input that causes the system to acceptably track the desired trajectory (dashed) around a set of slalom poles. This figure shows one of 10 such slalom courses, which were used as training data for the WLT algorithm. The problem addressed by this paper and by the WLT method, is the estimation of the blue input trajectory, given a previously unseen desired trajectory, and knowledge of previously flown trajectories.

constant component to the window matrix, or learning of offsets from the desired trajectory; however tests showed little improvement when these extensions were introduced.

Also note that the window matrix is constructed from actual output trajectories s^i. This contradicts (6), where the desired trajectories $\bar{s}^i$ were used. The quadrocopter slalom system used for these experiments terminates learning once the quadrocopter can fly successfully through the slalom course, rather than waiting for convergence to the desired trajectory. The implication of this is that accepted flights do not always accurately track the desired trajectory, thus in order to better capture the system dynamics, s^i is used in the learning process.

For the purposes of this experiment, the window horizon parameter was selected as $H = 50$, implying that each window $\mathcal{W}(\cdot)$ and parameter vector has size 101. With this selection, one second of historical data, and one second of future data (for both x and y trajectories) is included within the window, allowing for the system's delayed dynamics to be captured by the fit. Larger horizons were tested, but showed little benefit.

C. Summary of experiments

Using the experimental setup presented in Subsection III-A, 13 separate slaloms were learned, each using a different placement of slalom poles. For each learned slalom, the output trajectory s^i and the IILC learned input trajectory u^i are recorded. Both trajectories consist of x and y coordinates. One example of a slalom course learned by the IILC system is shown in Fig. 3.

The 13 learned slaloms were divided into two sets, with 10 slaloms used for training, and the remaining three used as test data, presented in Section IV. Training data comprised four three-pole and six four-pole slaloms, while test data comprised two three-pole, and one four-pole slalom.

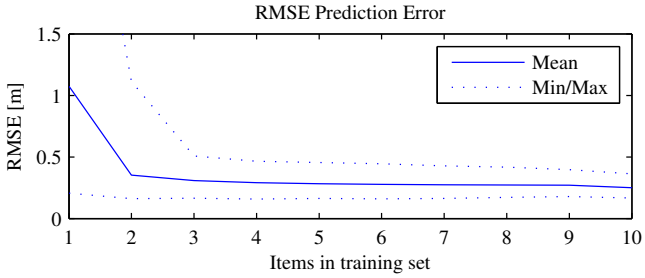

Fig. 4. Ten slaloms were learned and WLT parameters were trained using all possible combinations of these learned slaloms. These trained parameters were then used to predict the results of the slalom courses that were not included in the respective combination. This figure shows the prediction errors made for a varying number of items in the training set. The full set of 10 slaloms was used to predict every learned slalom (although all were included in the training set). This value represents the best error reduction possible using the algorithm when applied to the training set. Also apparent in this figure is the over-fit when only one trajectory is used for training.

D. Transferability of Learning

To investigate the transferability of learning between slaloms, we took the training set of 10 trajectories (introduced in Subsection III-C) and, beginning with one item in the training set and progressing through to a full training set of 10 trajectories, generated all ($\sum_{r=1}^{10} {}_{10}\mathcal{C}_r = 1023$) possible combinations of training sets. A separate WLT parameter vector was then trained for each combination and was used to predict the slaloms that were not included in the respective combination of the training set. The full training set of 10 trajectories was used to predict all slaloms and represents the best error reduction achievable using the WLT algorithm when applied to this training set.

For example, for training sets consisting of six priors, ${}_{10}\mathcal{C}_6 = 210$ combinations were generated. One such combination is $\{1, 4, 5, 6, 7, 8\}$, which was then used to predict the remaining four slaloms, $\{2, 3, 9, 10\}$.

Prediction accuracy was measured using the Root Mean Squared Error (RMSE) between the predicted and the IILC-learned input trajectory. Results of the aforementioned investigation are shown in Fig. 4. Also apparent is the over-fit for small training sets.

E. Mitigation of Overfitting

As apparent in Fig. 4, the proposed WLT algorithm overfits the data if only a single training trajectory is used for training. This could be mitigated by only applying the WLT algorithm after two or more slaloms have been learned by the IILC process. Another approach to mitigating over-fit is parameter reduction.

As discussed in Subsection III-B, a window horizon of $H = 50$ was used, which resulted in a window matrix with 202 parameters. Using the full training set, parameter redundancy was investigated by iteratively removing the lowest magnitude parameter from the window matrix and recalculating the least-squares fit for the reduced window. For each iteration, the RMSE error along all training trajectories was recalculated. This analysis found that a subset of nine parameters resulted in a prediction RMSE of 18.9cm, in

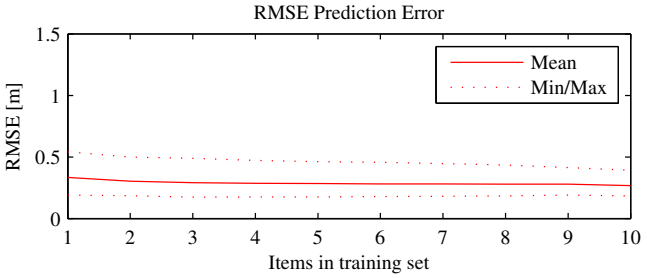

Fig. 5. This figure draws a comparison with Fig. 4, and shows the performance of the WLT algorithm when the principal 9 components from 202 are selected. This 4.5% accounts for 93% of full-set training performance, while resulting in improved performance for smaller training sets. Parameter reduction solves overfitting issues identified in Subsection III-D, and allows prediction after only a single slalom has been learned.

comparison to an RMSE of 17.6cm when the full window was used – 93% of performance is given by only 4.5% of the parameters.

The location of these parameters shows a dependence in θ_{xx} and θ_{yy} on both prior and future states ± 0.5 seconds from the current state, and a cross-coupled dependence in both θ_{xy} and θ_{yx} on future states 1 second from the current state. This non-causal dependence is not a problem, as trajectories are precalculated and supplied open-loop to the system's controller (Subsection III-A).

Using these nine parameters, the analysis presented in Subsection III-D was repeated. This yielded the results shown in Fig. 5. A comparison to Fig. 4 shows that overfitting issues have been significantly reduced and that the prediction performance remains approximately constant, allowing prediction after only a single slalom has been learned. Improvements to prediction are still possible by training from multiple trajectories, although these are relatively minor.

F. Prediction Accuracy

Using the nine parameters identified in the last subsection, the WLT algorithm's prediction ability was tested. A separate WLT parameter vector was trained for each of the 10 slaloms in the training set. These trained parameter vectors were subsequently used to predict the other nine trajectories in the training set. Example results of this prediction are shown in Fig. 6.

These results show that the primary dynamics of the input trajectory can be reconstructed from the desired trajectory, however subtleties such as higher frequency dynamics, which differ between trajectories, are not captured well by the model. Errors that appear towards the end of the trajectory are a result of the assumption that the trajectory ends at standstill – a condition not enforced by the slalom IILC algorithm [10].

IV. Experimental Results

In this section, we present the performance gains achievable by initializing an IILC system with a predicted input, as opposed to the nominal input, which is calculated using nominal model dynamics and an assumption of zero exogenous disturbances.

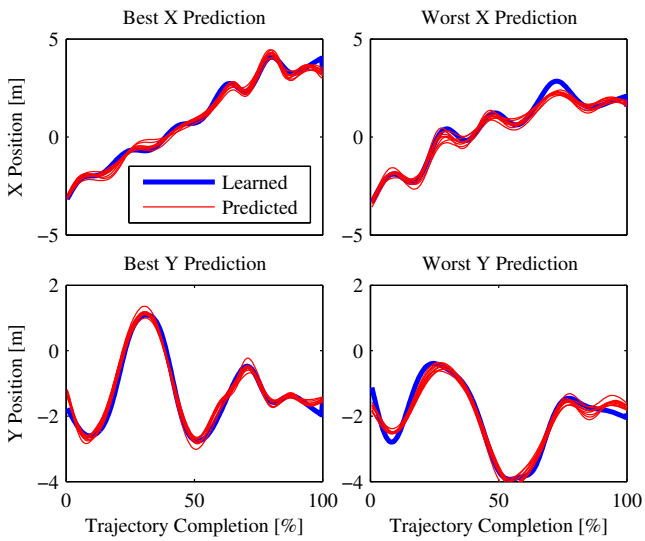

Fig. 6. This figure shows the ability of the WLT algorithm to predict unseen slalom trajectories, given a desired trajectory. Two trajectories from the training set are plotted column-wise, with x and y trajectory components plotted in the first and second row, respectively. For each trajectory, the IILC-learned input trajectories are shown in thick blue. The red lines show the various predictions made by a WLT method, which was trained on a single learned slalom from the training set. Optimally, the predicted reference inputs (red) would perfectly trace the IILC-learned input (blue). This figure shows that the major dynamics of the input trajectory are captured by the algorithm.

As discussed in Subsection III-F and shown in Fig. 6, knowledge transfer is possible after a single slalom has been learned, if the discussed parameter reduction is implemented. To test this, three unseen slalom courses were constructed. Two of these, referred to as "four pole" and "straight 3", had dynamics resembling the training set. The third unseen trajectory, "aggressive 3", was designed to require more aggressive dynamics.

A single prior slalom was then selected from the training set and was used to predict the required input for each unseen slalom using WLT. The unseen course was then flown using the WLT-predicted input and the deviation from the desired trajectory was recorded. This experiment was repeated for each of the 10 prior slaloms in the training set, each time using only a single prior to predict the three unseen courses. The results of these experiments are summarized in Table I.

Two example experiments are shown in Fig. 7: the flights corresponding to the prior with the lowest RMS prediction error across all three slaloms, and those corresponding to the prior with the highest. This figure and the results summarized in Table I show that a single prior trajectory can be used to accurately predict multiple unseen trajectories, with accuracy of prediction dependent on similarity of dynamics.

V. Conclusions & Outlook

The method for learning transfer between IILC-learned trajectories proposed in this paper passes a time-window over each point of the desired trajectory and uses a linear least-squares fit over this window to predict the required input at the respective time step. By considering previous and future

TABLE I

A COMPARISON OF THE INITIAL TRACKING PERFORMANCE OF AN UNSEEN SLALOM COURSE WHEN INITIALIZED USING WLT. THESE RESULTS ARE SHOWN GRAPHICALLY IN FIG. 7.

Slalom	**Four Pole**	**Straight 3**	**Aggressive 3**
Benchmark RMSE [m]			
Nominal	1.53	1.43	1.46
Converged	0.07	0.06	0.14
Single-Prior WLT Initialization RMSE [m]			
Mean	0.15	0.17	0.28
Std	0.05	0.07	0.08
Min	0.09	0.07	0.20
Max	0.24	0.29	0.45

states, the method is capable of modeling system delays and non-unit gains.

As shown in Fig. 6 and Fig. 7, the Windowed Learning Transfer (WLT) method is capable of learning from previous IILC results and of using this knowledge to predict the required input trajectory for an unseen trajectory. Furthermore, by reducing parameters to a primary subset (Subsection III-E), only a single prior trajectory is required for order-of-magnitude performance gains; our experimental results showed that using a single prior to predict and initialize the input for an unseen slalom course decreased initial tracking error (Fig. 7) from an average RMS tracking error of 1.48m (when initialized nominally), to 0.20m (when initialized with a WLT prediction trained on a single trajectory).

However as shown in Fig. 6, WLT captures only the dynamics transmitted through the system with a high gain. Using a more complex (than linear) mapping function may be able to capture these dynamics, and thus improve performance. A further limitation of the WLT method is also its naivety. The method ignores nominal model knowledge and the full state trajectory, which could be incorporated into the window matrix to potentially improve prediction accuracy. Finally, it is unclear to what degree the primary parameter subset (identified in Subsection III-E) can be extended to other trajectories. These questions are left open for further investigation.

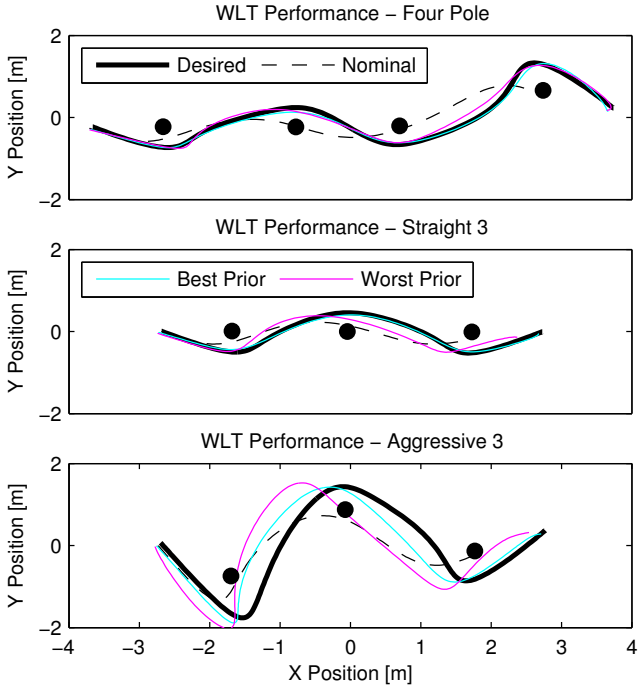

Fig. 7. Three unseen slalom courses were constructed and a single prior slalom was selected from the training set. This single prior was used to predict and initialize all three unseen slalom courses using WLT. This process was repeated for all 10 priors in the training set. Two example experiments are shown in this figure: the flights corresponding to the prior with the lowest RMS prediction error across all three slaloms (cyan), and the flights corresponding to the prior with the highest (magenta). Tracking performance can be compared to the desired trajectory, shown in thick black, and to the results of a nominal initialization, shown in thin dashed black. Optimally, the cyan and magenta WLT trajectories should closely track the thick-black desired trajectory.

REFERENCES

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of Robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, Jan. 1984.

[2] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, Jun. 2006.

[3] Y. Wang, F. Gao, and F. J. Doyle, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, Dec. 2009.

[4] M. Arif, T. Ishihara, and H. Inooka, "Incorporation of experience in iterative learning controllers using locally weighted learning," *Automatica*, vol. 37, pp. 881–888, 2001.

[5] ——, "Generalization of iterative learning control for multiple desired trajectories in robotic systems," *PRICAI 2002: Trends in Artificial Intelligence*, pp. 295–304, 2002.

[6] ——, "Intelligent learning controllers for dynamic non-linear systems using neural networks," in *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, vol. 5, no. 2. Soc. Instrument & Control Eng. (SICE), 2002, pp. 2782–2787.

[7] S. Gopinath, I. Kar, and R. Bhatt, "Experience inclusion in iterative learning controllers: Fuzzy model based approaches," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 4, pp. 578–590, Jun. 2008.

[8] P. Janssens, G. Pipeleers, and J. Swevers, "Initialization of ILC based on a previously learned trajectory," in *Proceedings of the 2012 American Control Conference*, 2012, pp. 610–614.

[9] J. Asensio, W. Chen, and M. Tomizuka, "Robot Learning Control Based on Neural Network Prediction," in *Proceedings of the 2012 ASME Dynamic Systems and Control Conference*, 2012, pp. 1489–1497.

[10] F. L. Mueller, A. P. Schoellig, and R. D'Andrea, "Iterative learning of feed-forward corrections for high-performance tracking," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 3276–3281.

[11] A. P. Schoellig and R. DAndrea, "Optimization-Based Iterative Learning Control for Trajectory Tracking," in *European Control Conference*, vol. 33, no. 1-2, Budapest, Hungary, Apr. 2009, pp. 1505–1510.