Trajectory generation for networked UAVs using online learning for delay compensation

Jaehyun Yoo, Seungjae Lee, H. Jin Kim, and Karl H. Johansson

Abstract— **This paper presents a trajectory generation mechanism based on machine learning for a network of unmanned aerial vehicles (UAVs). For delay compensation, we apply an online regression technique to learn a pattern of network-induced effects on UAV maneuvers. Due to online learning, the control system not only adapts to changes to the environment, but also maintains a fixed amount of training data. The proposed algorithm is evaluated on a collaborative trajectory tracking task for two UAVs. Improved tracking is achieved in comparison to a conventional linear compensation algorithm.**

I. INTRODUCTION

Time-delay compensation is important problem for networked control systems because varying delays induced by the network are known to degrade control performance. This paper considers trajectory generation for networked unmanned aerial vehicles (UAVs) such systems are highly delay sensitive in that they could be easily damaged if sensor information or control actuations are delayed.

There is a vast literature on time-delay control system. In [1], [2], predictive control approaches are proposed to provide a local plant with a sequence of future control inputs, where dynamic model of plants are known so that compensation is done easily by sending time-stamps on communication packet. However, such models and time-invariant delay assumption are seldom available in practice, such as the UAV scenario considered in this paper. In [3], [4], adaptive control schemes have been also suggested, where the assumption of constant delay or known delay function is required. In [5]–[7], they focus on designing low-level controllers to achieve maximum allowable delay and to hold stability, which have different goal to our path planning problem.

The main contribution of this paper is to apply a machine learning technique for multi-robot system with time-varying delay. It does not only require to put restrictions on the time-delay such as a known upper bound or being constant, but also need a known dynamic model. Therefore, in this paper, the role of the machine learning supports the lack of the dynamic model information and compensates for time-varying delays.

Many works have reported enhancement of control systems by applying machine learning techniques [8]–[10]. However, they focused on application of batch learning where the model learnt in the training phase does not adapt to changes to the environment. For the multi-UAV case with time-varying delay, those batch learning methods are not suitable.

In this paper, the application of an online machine learning is suggested, which updates a learning model efficiently whenever new training data is available without training all data from a scratch. It maintains a fixed amount of training data, which lessens computational load. We employ online square support vector regression (online LS-SVR) [11]. This algorithm has a beneficial characteristics of sparsity in which only a few training points called support vectors represent the learned model, while most of the other training samples become meaningless after training. The support vectors are exploited to extend a batch LS-SVR to an onliner version. Online LS-SVR has been employed in many practical applications such as robotics and sensor networks [12], [13].

As a robotic platform for which the suggested algorithm is applied, we consider one centralized trajectory planning for distributed UAVs that are networked in far distance. There are uplink and downlink channels between them. The uplink delay is dealt with model predictive control (MPC) method that provides a trajectory sequence including predictive future trajectories. The UAVs choose a proper input among them corresponding to uplink network condition. The downlink delay is compensated by the online learning that estimates UAV's state given delayed observation.

The proposed algorithm is evaluated to collaborative trajectory tracking of two UAVs. Simulations are performed for variable delays and tracking scenarios. Improved tracking results of the learning-based compensation are achieved in comparison to a conventional linear compensation method. Also, we report adaptive tracking performances against randomly varying time-delay.

The rest of this paper is organized as follows. Section II overviews the networked multi-UAV system and compensation methods for delays on uplink and downlink channels. Section III introduces a model predictive control (MPC), and Sections IV and V describe an online learning and its application to delay compensation, respectively. Sections VI and VII present simulation results and concluding remarks.

II. SYSTEM OVERVIEW

UAVs are time-sensitive in that they could be easily damaged if communication delay lasts over a tolerable period. For delay compensation of the uplink channel, we apply model predictive control (MPC) to provide control input sequence with the UAVs to maintain persistent control inputs.

*This work has been supported in part by the Knut and Alice Wallenberg Foundation, the Swedish Research Council (VR), and the Swedish Foundation for Strategic Research, including the SSF-NRF Sweden-Korea research program.

J. Yoo and K. H. Johansson are with the School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden, Emails: {jaehyun,kallej@kth.se}

S. Lee and H. Jin Kim are with the Department of Mechanical & Aerospace Engineering, Seoul National University, Seoul, South Korea, Emails: {sjlazza,hjinkim@snu.ac.kr}

The delay compensation of the downlink channel can be seen as an estimation problem. Given delayed observation, it aims to predict UAV state at current time step. This paper suggests a machine learning based estimation.

In this section, we first describe problem formulation about trajectory planning for networked UAVs in Section II-A. Then, Sections II-B and II-C overview the delay compensation methods on each of the uplink and the downlink channels.

A. Trajectory generation for networked UAVs

This paper considers a quadrotor UAV system. Let us define a state $\bar{x} = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \phi, \theta, \psi]^T$, where $[\mathrm{x}, \mathrm{y}, \mathrm{z}]$ is position of UAV in the inertial frame, and $[\phi, \theta, \psi]$ is roll, pitch, and yaw angles in the body frame. Many existing works such as [14], [15] have developed successful controller for quadrotors. Given a nonlinear dynamic model

$$\ddot{\bar{x}} = f(\bar{x}, \bar{u}), \quad (1)$$

they derive control inputs such that $\|[\mathrm{x}, \mathrm{y}, \mathrm{z}]^T - x^*(k)\| \to 0$, where $x^*(k)$ is a dynamic trajectory reference.

Our objective is to design a trajectory generator for $x^*(k)$ at the central side. Existence of random time-delay is the challenge to this problem. We assume that

- Local UAVs have individual position and attitude controllers, to track a dynamic trajectory reference provided by an central server.
- A trajectory planner does not know the dynamic models and the controllers of the local UAVs.
- The trajectory generation focuses on x-y position.

By the assumptions, we define symbol of state x to include 2-D position only, so we can define $x(k) = [\mathrm{x}(k), \mathrm{y}(k)]^T$. Taking the assumptions into account, we consider a trajectory generation model:

$$x(k+1) = Ax(k) + Bu(k), \quad (2)$$

and the goal is to derive control input u_k^* such that $\|x(k) - r(k)\| \to 0$, where $r(k)$ is a given reference. As a result, resultant u_k^* becomes a dynamic trajectory reference input to the local UAVs.

Note that we do not know A and B. Instead, this paper simplifies the model such that

$$A = I_{2\times 2} \quad \text{and} \quad B = vI_{2\times 2} \quad (3)$$

with the identity matrix $I_{2\times 2}$ and a constant v.

B. Uplink delay compensation

Suppose that the trajectory generator sends a sequence of control inputs $u(k+j-1|k)$ for $j = 1, \ldots, N_u$ to an UAV at time step k. The received packet is stored in a buffer at the UAV. To compensate for the uplink delay, control input $(\lfloor f_l \cdot \tau_{u,k} \rceil + 1)$ from the latest control sequence is selected, where f_l is the control frequency, $\tau_{u,k}$ is the uplink delay, and $\lfloor \cdot \rceil$ denotes the nearest integer. The buffer compares time-stamps of a newly arrived packet and the existing packet, and then it keeps the sequence having the latest time-stamp.

C. Downlink delay compensation

Let $y(k|k - \tau_{d,k})$ be the full-state observation sent from an UAV, where $\tau_{d,k}$ is the downlink delay. Given the delayed observation, we can predict the state of an UAV at time step k, by the following equation:

$$\hat{x}(k) \;=\; y(k|k - \tau_{d,k}) + \sum_{j=0}^{\lfloor f_l \cdot \tau_{d,k} \rceil - 1} A^j B u(k-1-j). \quad (4)$$

As long as we know the matrices A and B, (4) accurately compensates for the downlink time-delay. However, it is difficult to know A and B exactly in case of the UAV system due to its nonlinearity. For disturbance compensation, machine learning supports lack of the dynamic model information. Detail of the learning-based compensation is shown in Section V.

III. MPC for networked UAV control

MPC is a control strategy that calculates predictions of current and future control inputs by solving a constrained optimal control problem over a finite time horizon.

Suppose that $x_q \in \mathcal{R}^n$ be a state of the q-th UAV among N UAVs, and $x = \left[x_1^T, \ldots, x_N^T\right]^T$ be the concatenated state. Similarly, we define $r \in \mathcal{R}^{nN}$ and $u \in \mathcal{R}^{mN}$ as the reference and the control input vectors, respectively. Taking the downlink time-delay $\tau_{d,k}$ into account, the MPC optimization is given by:

$$\min_{u(k+j-1|k),\ j\geq 1} J_k = \sum_{i=1}^{N_p} \|x(k+i) - r(k+i)\|_Q^2 + \sum_{j=1}^{N_u} \|\Delta u(k+j-1)\|_R^2,$$

subject to

$$\begin{aligned}
x(k+i) &= Ax(k+i-1) + Bu(k+i-1), \\
x(k) &= [\hat{x}_1^T(k), \ldots, \hat{x}_N^T(k)]^T, \\
\hat{x}_q(k) &= g\left(y_q(k|k - \tau_{d,k}^q)\right) \text{ for } q = 1, \ldots, N, \quad (5)\\
x_{\min} &\leq x(k+i-1) \leq x_{\max}, \\
\Delta u_{\min} &\leq \Delta u(k+j-1) \leq \Delta u_{\max}, \\
u_{\min} &\leq u(k+j-1) \leq u_{\max}, \\
i &= 1, \ldots, N_p, \\
j &= 1, \ldots, N_u,
\end{aligned}$$

where N_p and N_u are lengths of prediction and control horizons, respectively. The matrices Q and R are constant weighting matrices, and the vectors $\Delta u_{\min}$, $\Delta u_{\max}$, $u_{\min}$, $u_{\max}$, $x_{\min}$, and $x_{\max}$ are constant constraint vectors.

The optimization results in the control inputs $u^*(k+j-1|k)$ over the control horizon $j = 1, \ldots, N_u$. The q-th control input $u_q^*(k+j-1|k)$ is sent to the q-th UAV through the uplink channel. The MPC optimization runs once the initial states over all UAVs arrive at the central controller. The initial state $\hat{x}_q$ in (5) is estimated by a function based on the delayed observation y_q, which is sent through the downlink channel.

Because the time delay $\tau_{d,k}$ may substantially degrade control performances, this paper focuses on modelling the function $g(\cdot)$. In order to obtain a more accurate model than a deterministic compensation such as (4), the following section introduces a machine learning method.

IV. Online LS-SVR

In this section, we first present the batch LS-SVR, and then the extension to the online version in Section IV-A.

The basic idea of LS-SVR is to map the data $\mathbf{x} \in \mathbb{R}^M$ to a higher dimensional feature space $\mathcal{H}$ (reproducing kernel Hilbert space) using a nonlinear mapping $\phi(\mathbf{x}) : \mathbb{R}^M \to \mathcal{H} \in \mathbb{R}^h$, and then find the relationship between scalar target variable y and explanatory variable $\mathbf{x}$ (i.e., linear regression) in the kernel space. In other words, given a training set of l training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ where $y_i \in \mathbb{R}$ are labels of the training data, it maps the training samples to a new data set $\{(\phi(\mathbf{x}_i), y_i)\}_{i=1}^{l}$ with the nonlinear mapping $\phi(\cdot)$.

Consider the linear regression model:

$$f(\mathbf{x}) = \langle w, \phi(\mathbf{x})\rangle + b, \quad w \in \mathbb{R}^h,\ b \in \mathbb{R},$$

where w and b are the coefficients, which are estimated by the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \frac{1}{2}\|w\|^2 + c\frac{1}{2}\sum_{i=1}^{l}\xi_i^2, \\ \text{s.t.} \quad \xi_i = y_i - f(\mathbf{x}_i), \end{aligned}$$

where the constant $c > 0$ is weight parameter.

We have a Lagrangian to solve the optimization given by:

$$\begin{aligned} L(w,b,\xi;\alpha) &= \frac{1}{2}w^{\mathrm{T}}w + \frac{1}{2}c\sum_{i=1}^{l}\xi_i^2 \\ &- \sum_{i=1}^{l}\alpha_i\left\{y_i - \left(w^{\mathrm{T}}\phi(\mathbf{x}_i) + b\right) - \xi_i\right\}, \end{aligned}$$

where $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers. The optimization conditions are as follows:

$$\begin{aligned} \frac{\partial L}{\partial w} &= 0 \to w = \sum_{i=1}^{l}\alpha_i\phi(\mathbf{x}_\mathrm{i}), \\ \frac{\partial L}{\partial b} &= 0 \to \sum_{i=1}^{l}\alpha_i = 0, \\ \frac{\partial L}{\partial \xi_i} &= 0 \to \alpha_i = c\xi_i, \\ \frac{\partial L}{\partial \alpha_i} &= 0 \to y_i - \left(w^{\mathrm{T}}\phi(\mathbf{x}_i) + b\right) - \xi_i. \end{aligned}$$

After elimination of the variables w and e, the solution is given by the set of linear equation:

$$\mathbf{A}_l\boldsymbol{\alpha}_l = \mathbf{y}_l, \tag{6}$$

with

$$\begin{aligned} \mathbf{A}_l &= \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & k(\mathbf{x}_1,\mathbf{x}_1) + \frac{1}{c} & \cdots & k(\mathbf{x}_1,\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k(\mathbf{x}_l,\mathbf{x}_1) + \frac{1}{c} & \cdots & k(\mathbf{x}_l,\mathbf{x}_l) + \frac{1}{c} \end{bmatrix}, \\ \boldsymbol{\alpha}_l &= \begin{bmatrix} b \\ \alpha_1 \\ \vdots \\ \alpha_l \end{bmatrix}, \quad \mathbf{y}_l = \begin{bmatrix} 0 & y_1 & \cdots & y_l \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{7}$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j)$ for $i, j = 1, \ldots, l$ is a kernel function. Finally, by Mercer's theorem, the fitting function as the output of LS-SVR is given by:

$$f(\mathbf{x}) = \sum_{i=1}^{l}\alpha_i k(\mathbf{x}, \mathbf{x}_i) + b,$$

where α_i and b are obtained from solution of linear equation in (7).

In this paper, the kernel function is defined by radial basis function (RBF):

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(c_1 \cdot \|\mathbf{x}_i - \mathbf{x}_j\|/c_2\right),$$

where c_1 and c_2 are weight parameters to control strength and smoothness of the kernel function. Therefore, the learning output is rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^{l}\alpha_i \exp\left(c_1 \cdot \|\mathbf{x}_i - \mathbf{x}\|/c_2\right) + b. \tag{8}$$

A. Online LS-SVR based on incremental and decremental algorithms

As a new data point arrives, the online algorithm using incremental update should improve the previous model with a smaller effort than the batch implementation. Also, the decremental algorithm safely removes worthless or less important data in order to limit the data size. In this section, we extend the batch LS-SVR from (6) to the online version by following incremental and decremental algorithms.

1) Incremental Algorithm: The incremental algorithm updates the pre-trained LS-SVR when a new sample $(\mathbf{x}_{l+1}, y_{l+1})$ is added to the existing training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$. From (6), the incremental relationship between the current model and the new model is as follows:

$$\mathbf{A}_{l+1}\boldsymbol{\alpha}_{l+1} = \mathbf{y}_{l+1}, \tag{9}$$

with

$$\mathbf{A}_{l+1} = \begin{bmatrix} \mathbf{A}_l & \mathbf{a}_l \\ \mathbf{a}_l^T & C_l \end{bmatrix}, \quad \boldsymbol{\alpha}_{l+1} = \begin{bmatrix} \boldsymbol{\alpha}_l \\ \alpha_{l+1} \end{bmatrix}, \quad \mathbf{y}_{l+1} = \begin{bmatrix} \mathbf{y}_l \\ y_{l+1} \end{bmatrix},$$

and

$$\begin{aligned} \mathbf{a}_l &= [1, k(\mathbf{x}_1, \mathbf{x}_{l+1}), \ldots, k(\mathbf{x}_l, \mathbf{x}_{l+1})]^T, \\ C_l &= 1/c + k(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}). \end{aligned}$$

The online incremental training algorithm aims to efficiently update $\mathbf{A}_{l+1}^{-1}$ whenever a new sample arrives, without computation of the matrix inverse. By the block matrix inversion lemma, $\mathbf{A}_{l+1}^{-1}$ can be given by:

$$\mathbf{A}_{l+1}^{-1} = \begin{bmatrix} \mathbf{A}_l^{-1} + d_l \mathbf{A}_l^{-1}\mathbf{a}_l \mathbf{a}_l^T \mathbf{A}_l^{-1} & d_l - \mathbf{A}_l^{-1}\mathbf{a}_l \\ -\mathbf{a}_l^T \mathbf{A}_l^{-1} & d_l \end{bmatrix},$$

with $d_l = (c - \mathbf{a}_l^T \mathbf{A}_l^{-1}\mathbf{a}_l)^{-1}$.

2) Decremental algorithm: The purpose of the decremental algorithm is to remove the existing i-th sample (x_i, y_i) from training set when a sample is not important or amount of samples exceeds a limitation. The concept of the support vectors offers a natural criterion for the choice of the samples to be removed. Less important points have small $\|\alpha\|$, because the small α barely affects the learning output (8).

To avoid computing the matrix inverse, $\mathbf{A}_l^{-1}$ should be updated from $\mathbf{A}_{l+1}^{-1}$. Here, $\mathbf{A}_l^{-1}$ is the matrix without the k-th row and k-th column. Then, when the k-th sample is pruned from the $l+1$ pairs of the data set, the update is given by [16]:

$$a_{ij} \leftarrow a_{ij} - a_{kk}^{-1} a_{ik} a_{kj}, \tag{10}$$

where a_{ij} stands for the item at the i-th row and j-th column of $\mathbf{A}_{l+1}^{-1}$ for $i, j = 1, \ldots, l$; $i, j \neq k$, and k stands for the k-th training to be removed.

V. Delay compensation based on learning

In our strategy, the learning input is defined as a set of delayed observations and the linear compensation in (4). It compensates for the difference between the true state and the summation of the delayed observation and the linear compensation. Thus the estimate is given by:

$$\begin{aligned} \hat{x}(k) &= g\left(y(k|k - \tau_{d,k})\right) \qquad (11) \\ &= y(k|k-\tau_{d,k}) + \gamma_k + f(\{y(k|k-\tau_{d,k}), \gamma_k\}), \end{aligned}$$

where γ_k is the linear compensation term from (4):

$$\gamma_k = \sum_{j=0}^{\lfloor f_l \cdot \tau_{d,k} \rceil - 1} A^j B u(k-1-j),$$

and $f(\{y(k|k-\tau_{d,k}), \gamma_k\})$ is the output of the learning from (8). The training set $D = \{(\mathbf{x}_i, z_i)\}_{i=1}^{l}$ is defined as:

$$\begin{aligned} \mathbf{x}_i &= \{y_i, \gamma_i\}, \\ z_i &= x_i - (y_i + \gamma_i), \end{aligned}$$

where training input $\mathbf{x}_i$ is a set of the delayed observation and the linear compensation term. The training output is the residual between the true state x_i and the summation of the delayed observations and the deterministic compensation term. We note that, in the training phase, $x_i(k|k)$ are collected by the UAVs, while $y_i(k|k-\tau_{d,k})$ and γ_i are recorded in the central controller. The training is performed in the controller side after the data are gathered in the controller.

We emphasize that the combination of the linear compensation and the data-driven model via the machine learning is complementary in the compensation model in (11). At the early training phase when only a few training data points exist, it might produce an inaccurate learning model. This may generate improper control input. The combination of the linear compensation and the learning model prevents this situation because the linear compensation term γ_k is based on the deterministic model which is independent of the training data.

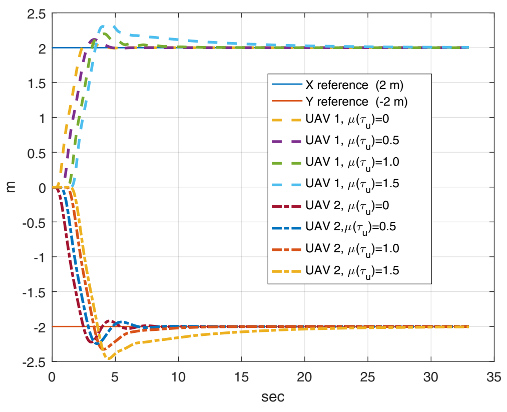


Fig. 1: Compensation for uplink delay τ_u that follows Gaussian distribution with fixed variance and variable mean: $\tau_u \sim N(\mu(\tau_u), 0.2)$.

VI. Simulation results

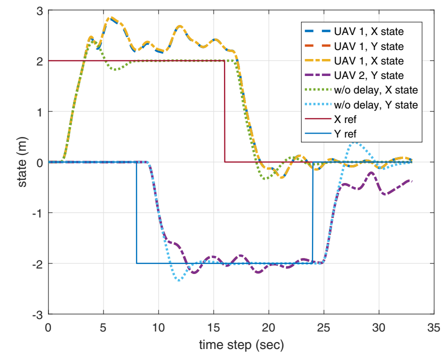
For the simulation study, we consider two nonlinear UAVs with different tracking performances, using the Matlab robotics toolbox developed by [17], which provides nonlinear UAV model simulink. We fix position and attitude controllers of the UAVs in advance, and add MPC trajectory generation with time-delay configuration in order to evaluate the suggested algorithm.

For MPC setup described in Section III, the constraints of the state and input are given by $|\text{x}|, |\text{y}| < 10$ and $|u_1|, |u_2| < 1$, and $v = 0.05$ in (3). The running cost function and terminal cost function are given by $Q = 2I_{2\times 2}$ and $R = 10I_{2\times 2}$. The prediction horizon is set to $N = 20$ steps and time sampling interval is set to 0.1 sec.

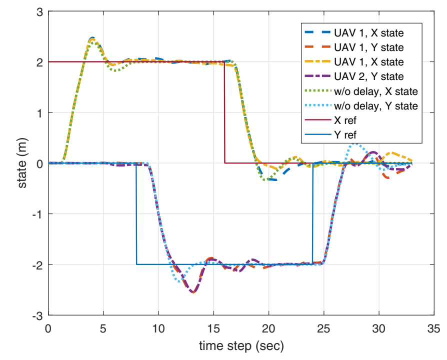
Simulation purpose is to let the UAVs track the same reference trajectory at the same time. Because there are random communication delays, the simultaneous tracking performance depends on how accurate the delay compensation is.

The result of the uplink delay compensation described in Section II-B is shown in Fig. 1, where we command the reference position (2,-2) to the two UAVs whose initial positions are (0,0). In this simulation, there is no downlink delay. As the uplink delay increases from 0 to 1.5 sec, the settling time and the overshoot response get larger. In all the following simulations, the uplink delay is fixed to have Gaussian distribution with mean 0.7 and variance 0.2. From now, simulations about downlink delay compensation will be examined.

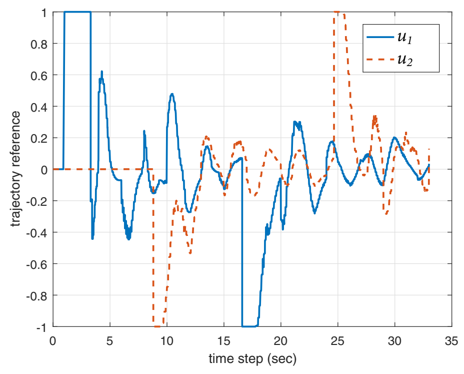
Fig. 2 compares the linear compensation and the (batch) learning based compensation for downlink delay that has a



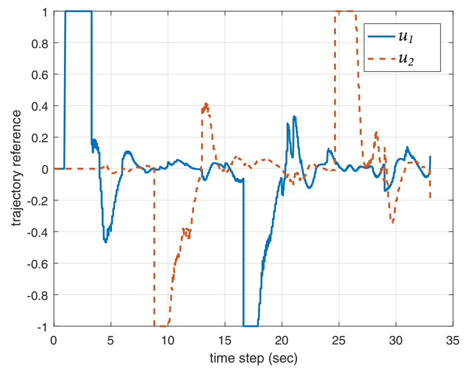
(a) Linear compensation



(b) Learning based compensation



(c) Linear compensation



(d) Learning based compensation

Fig. 2: Tracking results between the linear compensation and the batch learning based compensation, given waypoint reference.

Gaussian distribution with 2.5 mean and 0.2 variance, where the UAVs track waypoint reference in a finite time. Figs. 2(a) and 2(b) show tracking results of the linear and the learning compensations, respectively. From the results, we can see the better control performance of the suggested learning-based compensation. From Figs. 2(c) and 2(d), we can see the chattering control input of the compared algorithm, while the suggested algorithm yields more smooth control actions.

In order to evaluate a training performance in training phase, we define training error as:

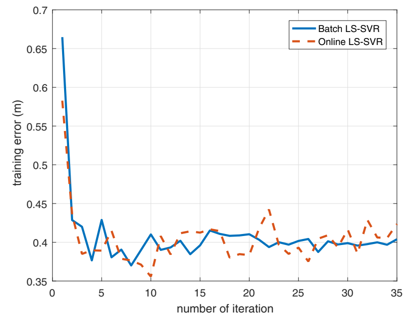
$$\text{Training error} = \frac{1}{m}\sum_{k=1}^{n}\sum_{j=1}^{m}\left(\|x_j(k) - r(k)\|\right),$$

where n is the number of iterations, m is the number of UAVs, x_j is the x-y state of the j-th UAV, and r is reference. Fig. 3 compares the batch learning and the online learning, where the step reference (2,-2) is given to the UAVs. Fig. 3(a) shows each training error of the batch learning and the online learning, and Fig. 3(b) draws the number of traning samples used for the learning at every iteration. In case of the online learning, 3000 training data is fixed after 10-th iteration step while the batch learning uses linearly increasing amount of training data. After training whose data, the online learning has 0.3982 m tracking error result in Fig. 3(c), and the batch learning has 0.3975 m tracking error, in test phase. From this result, we confirm that, in spite of amount of training data much smaller than data used for the batch learning, the online learning method provides tracking performance as accurate as the result of the batch LS-SVR.

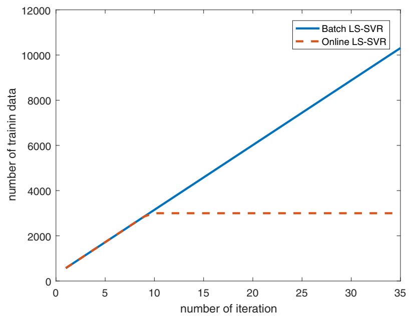
Finally, in order to evaluate adaptability of the online learning, we change distribution of the delay disturbance from $\tau_d \sim N(1.5, 0.2)$ to $\tau_d \sim N(2.5, 0.2)$ after 10-th training step as shown in Fig. 4(a). Since the training error had some spikes of the training error between 10$\sim$13 iterations, it has been kept stable under 0.4 m training error. That is, by 3 iterations, the online learning trains a new environmental data. After the training phase, we test two different situations whose results are shown in Figs. 4(b) and 4(c). Fig. 4(b) is the result when downlink delay is 1.5 sec and Fig. 4(c) is with 2.5 sec. From both results, we can see that the learning model can compensate for two different delay distributions, and that the results have performance as accurate as the result when the disturbance does not change as shown in Fig. 3(c).

VII. CONCLUSIONS

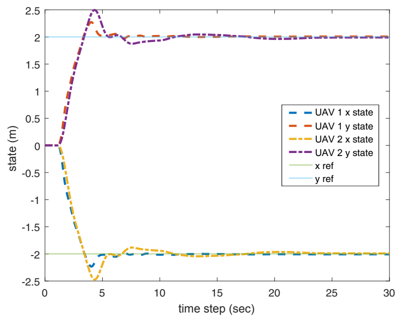
We considered a networked control for collaborative multi-UAV application, with compensation for random communication time-delays. The online learning based machine learning method was used to learn network-induced effects on UAV maneuvers, with adaptive compensation for variable time-delay. From simulation results, we found that the tracking results are improved in comparison to a deterministic compensation method and the batch learning due to the accurate and efficient online learning.



(a) Training error in training phase

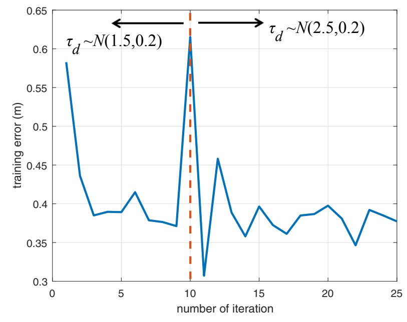


(b) The number of training data used in training phase

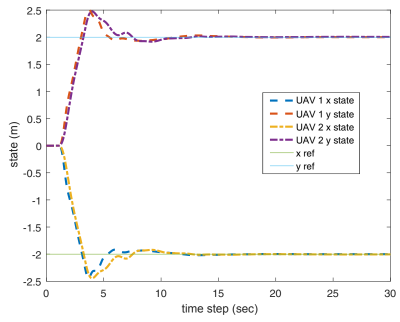


(c) Online learning (tracking error = 0.3982 m) in test phase

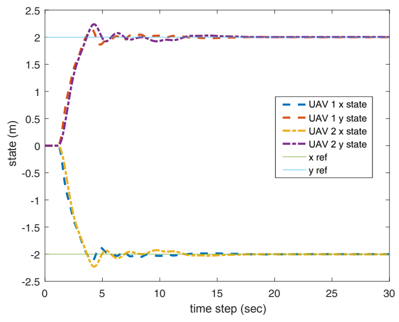
Fig. 3: Comparison between the batch learning and the online learning where (a): the number of used training data per iteration, (b): training error per training iteration, and (c): tracking result.



(a) Training error in training phase



(b) Online learning (tracking error = 0.3978 m) in test phase with $\tau_d \sim N(2.5, 0.2)$



(c) Online learning (tracking error = 0.3987 m) in test phase with $\tau_d \sim N(1.5, 0.2)$

Fig. 4: Adaptability of online learning according to change of delay distribution at 10-th iteration step, where (a): training error per training iteration, (b): tracking result with downlink delay $\tau_d \sim N(1.5, 0.2)$, and (c): tracking result with downlink delay $\tau_d \sim N(2.5, 0.2)$.

REFERENCES

[1] H. Zhang, Y. Shi, and M. Liu, "H∞ step tracking control for networked discrete-time nonlinear systems with integral and predictive actions," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 337–345, 2013.

[2] T. Zuo, H. Min, T. Zhang, and X. Zhang, "The self-tuning networked control system with online delay prediction," *Transactions of the Institute of Measurement and Control*, 2016.

[3] X. Jiao and T. Shen, "Adaptive feedback control of nonlinear time-delay systems: The lasalle-razumikhin-based approach," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1909–1913, 2005.

[4] S. S. Ge, F. Hong, and T. H. Lee, "Robust adaptive control of nonlinear systems with unknown time delays," *Automatica*, vol. 41, no. 7, pp. 1181–1190, 2005.

[5] P. Pepe and Z.-P. Jiang, "A lyapunov–krasovskii methodology for iss and iiss of time-delay systems," *Systems & Control Letters*, vol. 55, no. 12, pp. 1006–1014, 2006.

[6] M. Krstic, "Input delay compensation for forward complete and strict-feedforward nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 287–303, 2010.

[7] N. Bekiaris-Liberis and M. Krstic, "Compensation of state-dependent input delay for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 275–289, 2013.

[8] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *Decision and Control, IEEE International Conference on*. IEEE, 2016, pp. 4653–4660.

[9] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Robotics and Automation, IEEE International Conference on*. IEEE, 2016, pp. 491–496.

[10] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[11] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[12] J. Yoo and H. J. Kim, "Target localization in wireless sensor networks using online semi-supervised support vector regression," *Sensors*, vol. 15, no. 6, pp. 12 539–12 559, 2015.

[13] J. Yoo, W. Kim, and H. J. Kim, "Distributed estimation using online semi-supervised particle filter for mobile sensor networks," *IET Control Theory & Applications*, vol. 9, no. 3, pp. 418–427, 2015.

[14] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Robotics and Automation, IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4393–4398.

[15] H. Bou-Ammar, H. Voos, and W. Ertel, "Controller design for quadrotor UAVs using reinforcement learning," in *Control Applications, IEEE International Conference on*. IEEE, 2010, pp. 2130–2135.

[16] T. Poggio and G. Cauwenberghs, "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems*, vol. 13, p. 409, 2001.

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