Preprints of the 20th World Congress
The International Federation of Automatic Control
Toulouse, France, July 9-14, 2017

Learning communication delay patterns for remotely controlled UAV networks [⋆]

Jaehyun Yoo and Karl H. Johansson

Department of Automatic Control,
School of Electrical Engineering,
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: {jaehyun,kallej}@kth.se

Abstract: This paper deals with collaborative unmanned aerial vehicles (UAVs) that are remotely controlled from a cloud server. The main contribution is to apply machine learning technique to find a pattern of network-induced effects on maneuvers of UAVs, in order to compensate for time delays and packet losses in remote communication. As machine learning technique, a Gaussian process (GP) based approach is employed due to its computational simplicity and flexibility in modelling complex expressions using a small number of parameters. We combine a deterministic compensation for an enhanced GP model to overcome a problem of the lack of training data at the beginning of training phase. This is done by defining training data input as a set of delayed observation and the deterministic compensation term, and by training the GP on residual between the true state and the input set. The proposed algorithm is evaluated to collaborative trajectory tracking of two UAVs. Simulations are performed for various delays and tracking scenarios. It is shown that the better tracking results are achieved compared to a conventional linear compensation algorithm.

Keywords: Networked robotics, unmanned aerial vehicles, time-delay systems, machine learning.

1. INTRODUCTION

Small unmanned aerial vehicles (UAVs) have a great potential for remote applications such as traffic monitoring, inspection of hazard environments, and aerial transportation [Chao et al. (2008); Quaritsch et al. (2010)]. These applications become more useful as longer distances between the UAVs and a central control station are permitted. For maintaining stable flight performances at long distances, network-induced influences such as time delays and packet losses should be compensated, otherwise the UAVs may even become unstable.

To overcome delays and packet dropouts, many networked control systems (NCSs) approaches have been developed. In [Wang et al. (2010); Zhao et al. (2011); Onat et al. (2011)], predictive control approaches provide a local plant with a sequence of future control inputs. Then, the local plant chooses a proper control input corresponding to current network condition, where the plant is assumed to follow a linear dynamic model. Similarly, auto aggressive models in [Hu et al. (2007); Liu et al. (2006); Zuo et al. (2016)] and mixture of multiple models in [Jeong and Park (2005); Dong et al. (2010); Zhang et al. (2013)] are assumed to be known and used to generate control predictions. However, such models are seldom available

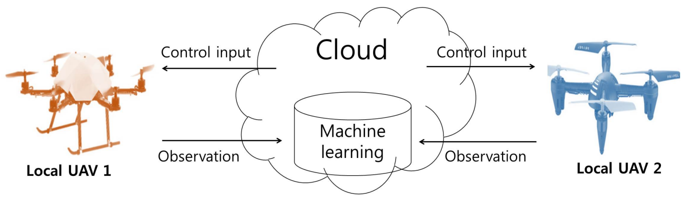


Fig. 1. Remotely controlled UAVs from a cloud server.

in practice, such as the UAV scenario considered in this paper.

The main contribution of this paper is to apply a machine learning technique to compensate for random communication delays, without requirement of UAV dynamic models. It does not learn time-delay itself, but learn a pattern of network-induced effects of UAV maneuvers. This is why we do not need precise dynamic models. Fig. 1 illustrates the system considered with remotely controlled UAVs from a cloud that implements a machine learning algorithm. Delayed observations sent from the UAVs are used as training data, and the cloud provides control inputs with the UAVs.

The machine learning technique we employ is a Gaussian process (GP) approach which has been recently used in many applications, due to its flexibility in modelling complex expressions using a small number of parameters. GP-based machine learning algorithms have been applied to aerial vehicles [Ko et al. (2007); Akametalu et al. (2014)] and sensor networks [Yoo et al. (2011); Osborne et al. (2008)] because of its computational simplicity. In

⋆ 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.

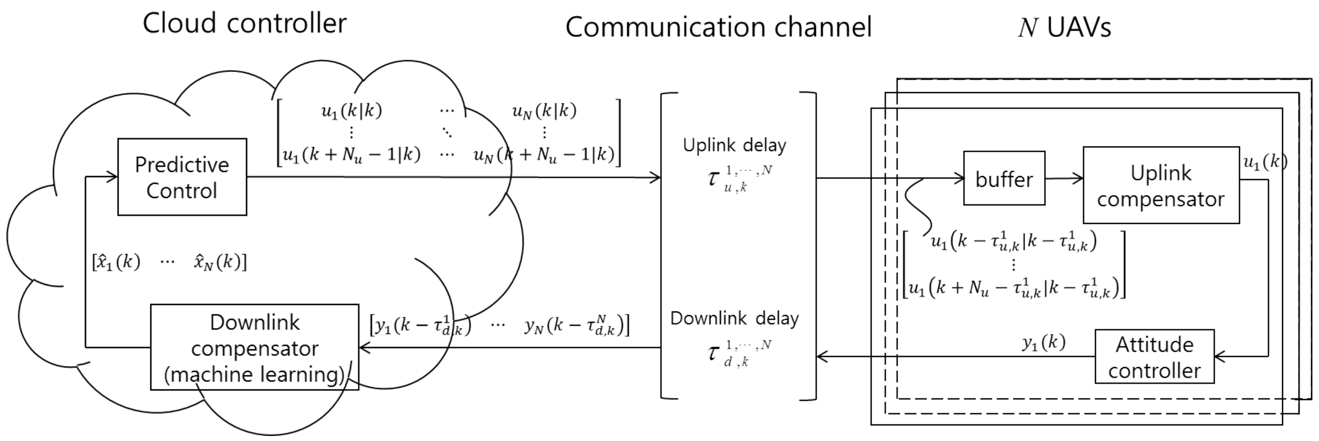

Cloud controller
Communication channel
N UAVs
Predictive Control
$\begin{bmatrix} u_1(k|k) & \cdots & u_N(k|k) \\ \vdots & \ddots & \vdots \\ u_1(k+N_u-1|k) & \cdots & u_N(k+N_u-1|k) \end{bmatrix}$
$[\hat{x}_1(k) \ \cdots \ \hat{x}_N(k)]$
Downlink compensator (machine learning)
$[y_1(k-\tau^1_{d,k}) \ \cdots \ y_N(k-\tau^N_{d,k})]$
Uplink delay $\tau^{1,\cdots,N}_{u,k}$
Downlink delay $\tau^{1,\cdots,N}_{d,k}$
buffer
Uplink compensator
$u_1(k)$
$\begin{bmatrix} u_1(k-\tau^1_{u,k}|k-\tau^1_{u,k}) \\ \vdots \\ u_1(k+N_u-\tau^1_{u,k}|k-\tau^1_{u,k}) \end{bmatrix}$
$y_1(k)$
Attitude controller

Fig. 2. Overview of networked predictive control with a machine learning compensation for remotely controlled UAVs.

comparison to other machine learning methods such as support vector machine, neural network, and deep learning [Li et al. (2015); Lv et al. (2016)], the GP approach is sometimes more efficient.

Standard GP model assumes that the process underlying training data is zero-mean. In case of our scenario, the zero-mean indicates that there is no difference between the true state and delayed state observation. It may generate improper control input and cause UAV crash because the UAVs are sensitive to even small change of control input. In other words, as long as we do not hold enough amount of training data (e.g., at the beginning of the training phase), we cannot help getting the wrong estimates. In order to overcome this problem, we combine a deterministic compensation model which is independent of training data and data-driven modelling to form an enhanced GP regression model. This is done by defining training data input as a set of delayed observation and the deterministic compensation term, and then by training the GP on residual between the true state and summation of the delayed observation and the deterministic compensation term.

The proposed algorithm is evaluated to collaborative trajectory tracking of multiple UAVs. Simulations are performed for various delays and tracking scenarios. It is shown that the better tracking results are achieved compared to a conventional linear compensation algorithm. We also analyze how the suggested approach improves the tracking performances.

The rest of this paper is organized as follows. Section 2 presents the networked UAV system as well as delay compensation scheme. Section 3 introduces a networked control scheme based on model predictive control (MPC). Section 4 describes the machine learning based compensation algorithm, and Section 5 shows simulation results. Finally, Section 6 is devoted to concluding remarks.

2. NETWORKED UAVS

Fig. 2 describes an overview of the networked predictive control scheme applied to UAVs. The networked UAVs are time-sensitive in that they could be easily damaged if communication delay lasts over a tolerable period. By applying the control scheme, the UAVs can maintain persistent control inputs compensating time-varying delay of the *uplink* channel. The delay compensation of the *downlink* channel can be seen as an estimation problem. Given delayed observation, the cloud can predict the UAV state at the current time. We suggest a machine learning based estimation scheme, which will be introduced in Section 4.

We first suppose the cloud controller and each UAV are time-synchronized and data packets are time-stamped. Second, upper bound of the time-varying network delay is not larger than N_1. Also, the number of the consecutive data dropouts is less than N_2. Therefore, the number of control predictions N_u is set to $N_u = N_1 + N_2$. Next, we describe the compensation methods for the uplink and the downlink delays in the following.

2.1 Uplink delay compensation

The cloud sends a sequence of control inputs $u_q(k+j-1|k)$ for $j = 1, \ldots, N_u$, to the q-th UAV at time step k. The received packet is stored in a buffer at the UAV. To compensate for the uplink delay, control input $(\lfloor f_l \cdot \tau^q_{u,k} \rceil + 1)$ 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 timestamps of a newly arrived packet and the existing packet, and then it keeps the sequence having the latest timestamp.

2.2 Downlink delay compensation

Let $y_q(k|k-\tau^q_{d,k})$ be the full-state observation sent from the q-th UAV, where $\tau^q_{d,k}$ is the downlink delay. Given the delayed observation, we can predict the current state of the q-th UAV at time step k, by the following equation:

$$x_q(k) = y_q(k|k-\tau^q_{d,k}) \sum_{j=0}^{\lfloor f_l \cdot \tau^q_{d,k} \rceil - 1} A_q^j B_q u(k - \lfloor f_l \cdot \tau^q_{d,k} \rceil - j). \qquad (1)$$

As long as we know the matrices A_q and B_q for all agents $q = 1, \cdots, N$, equation (1) accurately compensates for the downlink time delay. However, in this paper, we do not require precise dynamic models such as in[Lee et al. (2009)].

Instead, this paper simplifies the model to $A_q = I$ and $B_q = vI$ with the identity matrix I and velocity scalar v, where state and control input are defined as 2-D position and velocity, respectively. Machine learning supports the lack of the dynamic model information as well as compensates for the delays, by learning effects of the UAV maneuvers.

3. MODEL PREDICTIVE CONTROL (MPC) FOR UAV NETWORKS

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. By setting the number time horizon to the sum of the upper bounds of the delay and the dropout, i.e, $N_u = N_1 + N_2$, the MPC can compensate for the uplink delay.

Suppose that $x_q \in \mathcal{R}^n$ be a state of the q-th UAV among N UAVs, then the concatenated state $x = \left(x_1^T, \ldots, x_N^T\right)^T$, where $\cdot^T$ denotes the transpose. 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

$$\begin{aligned} x(k+i) &= Ax(k+i-1) + Bu(k+i-1), \\ x(k) &= [\hat{x}_1^T(k), \ldots, \hat{x}_N^T(k)]^T, \\ \hat{x}_q(k) &= f\left(y_q(k|k-\tau_{d,k}^q)\right) \text{ for } q = 1, \ldots, N, \\ x_{\min} &\leq x(k+i-1) \leq x_{\max}, \\ \Delta u_{\min} &\leq \Delta u(k+j-1) \leq \Delta u_{\max}, \\ u_{\min} &\leq u(k+j-1) \leq u_{\max}, \\ i &= 1, \ldots, N_p, \\ j &= 1, \ldots, N_u, \end{aligned} \tag{2}$$

where N_p and N_u ($N_p \geq N_u$) 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 cloud. The initial state of the q-th UAV in equation (2), i.e. $\hat{x}_q$, 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 $f(\cdot)$. In order to obtain more accurate model than a deterministic compensation such as equation (1), the following section introduces a machine learning method.

4. DELAY COMPENSATION USING GAUSSIAN PROCESSES LEARNING

This section describes a downlink delay compensation method using a GPs. Section 4.1 formalizes the GP algorithm and Section 4.2 applies the GP to the compensation.

4.1 GP learning

GP-based machine learning seeks posterior distributions over functions $g(\cdot)$ from training data $D = \{(\mathbf{x}_i, z_i)\}_{i=1}^l$ which is a set of l number of training data points drawn from

$$z_i = g(\mathbf{x}_i) + \varepsilon, \tag{3}$$

where the noise ε follows the Gaussian distribution $N(0, \sigma_{GP}^2)$ with zero mean and variance σ_{GP}^2. A key idea underlying GP is the requirement that the function values at different data points are correlated, where the covariance between two function values $g(\mathbf{x}_i)$ and $g(\mathbf{x}_j)$ depends on the input values $\mathbf{x}_i$ and $\mathbf{x}_j$. This dependency can be specified via a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. Selection of the kernel function depends on user's choice. This paper applies the Gaussian kernel function, given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta_1 \exp\left(\frac{-\parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2}{2\theta_2}\right), \tag{4}$$

where θ_1 is the signal variance and θ_2 is the length scale that determines how fast the correlation between data points decreases. The parameters represent the smoothness of the function estimated by the GP. The parameters can be learned by the conjugate gradient descent method [Rasmussen (2004)].

The joint distribution over the training outputs $\mathbf{z} = [z_1, \ldots, z_l]^T$ is a function of the training inputs $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_l]$ of the form

$$\mathbf{z} \sim N\left(0, K(\mathbf{X}, \mathbf{X}) + \sigma_{GP}^2 I\right), \tag{5}$$

where $K(\mathbf{X}, \mathbf{X})$ is an $l \times l$ kernel matrix whose (i, j)-th element is $k(\mathbf{x}_i, \mathbf{x}_j)$ in equation (4). For any set of values $\mathbf{X}$, one can generate the matrix K and then sample a set of corresponding targets $\mathbf{z}$.

After the training, at a point of interest point $\mathbf{x}_*$, the output estimate has a Gaussian distribution with mean $\mu_{\mathbf{x}_*}$ and variance $\sigma_{\mathbf{X}_*}^2$, given by:

$$p\left(g(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{z}\right) = N\left(g(\mathbf{x}_*); \mu_{\mathbf{X}_*}, \sigma_{\mathbf{X}_*}^2\right), \tag{6}$$

and

$$\begin{aligned} \mu_{\mathbf{x}_*} &= k_*^T \left(K(\mathbf{X}, \mathbf{X}) + \sigma_{GP}^2 I\right)^{-1} \mathbf{z} \\ \sigma_{\mathbf{x}_*}^2 &= k(\mathbf{x}_*, \mathbf{x}_*) - k_*^T \left(K(\mathbf{X}, \mathbf{X}) + \sigma_{GP}^2 I\right)^{-1} k_*, \end{aligned}$$

where k_* is the $l \times 1$ vector representing covariances between $\mathbf{x}_*$ and $\mathbf{X}$.

In addition to the mean value $\mu_{\mathbf{x}_*}$, the GP provides the variance $\sigma_{\mathbf{x}_*}^2$ at any interest, which is useful for a criterion to decide if a learning result is reliable. For example, [Krause et al. (2008)] studies a sensor network in which the GP variance is used to determine an optimal sensor placement. By locating sensor nodes at somewhere having large variance, it reduces uncertainty of sensor coverage. In [Berkenkamp et al. (2015)], the GP variance helps sampling the next position of an aerial robot. Safe control scheme is achieved by the variance information which does not allow the robot to act insecure motion.

4.2 Compensation

The GP compensation aims to find a relationship between the delayed observation $y(k|k-\tau_{d,k})$ and the ground truth state $x(k|k)$. In our strategy, the GP input is defined as a set of delayed observations and the linear compensation terms in equation (1). This setup is different to general setup where an input set of the GP is defined by the observation only, because the combination with the linear compensation can give better accuracy than the conventional GP approach using the raw observation only.

We compensate for the difference between the true state and the summation of the delayed observation and the linear compensation, given by:

$$f\left(y(k|k-\tau_{d,k})\right) = y(k|k-\tau_{d,k}) + \gamma_k + \mu_{\{y(k|k-\tau_{d,k}),\gamma_k\}}, \quad (7)$$

where γ_k is the linear compensation term:

$$\gamma_k = \sum_{j=0}^{\lfloor f_l \cdot \tau_{d,k} \rceil - 1} A^j B u(k - \lfloor f_l \cdot \tau_{d,k} \rceil - j), \quad (8)$$

and $\mu_{\{y(k|k-\tau_{d,k}),\gamma_k\}}$ is the mean estimate of the GP, where the training set $D = \{(\mathbf{x}_i, z_i)\}_{i=1}^{l}$ is defined as:

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

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 cloud. The training is performed after the data are collected in the cloud.

We emphasize that the combination of the linear compensation and the data-driven model via the GP is complementary. According to the GP prior model in equation (5), the output of a test sample tends to be zero when the input of the sample is not around the domain of the existing training data set. The zero output value as the GP estimate indicates that there is no difference between the the ground truth and the delayed observation, i.e., no error of the prediction. It may generate improper control input and cause UAV crash because the UAVs are sensitive to even small change of control input. In other words, as long as we do not hold enough amount of training data (e.g., at the beginning of the training phase), we cannot help getting the wrong estimates. However, because the linear compensation term γ_k obtained from the deterministic model is independent of the training data, the combination improves the prediction.

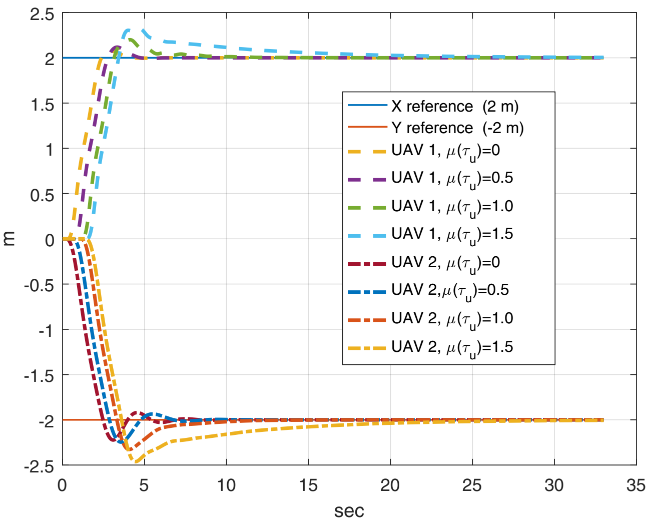


Fig. 3. Step responses of two UAVs according to variation of the uplink delay $\tau_u \sim N\left(\mu(\tau_u), 0.2\right)$

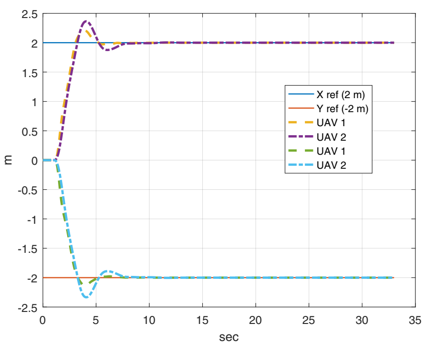
5. SIMULATION RESULTS

For the simulation study, we consider two nonlinear UAVs with different tracking performances, using the Matlab toolbox developed by Corke (1996). We fix attitude controllers of the UAVs in advance. The UAVs are to track the same reference trajectory simultaneously. Because there are random communication delay, the tracking performance depends on how accurate the delay compensation is.

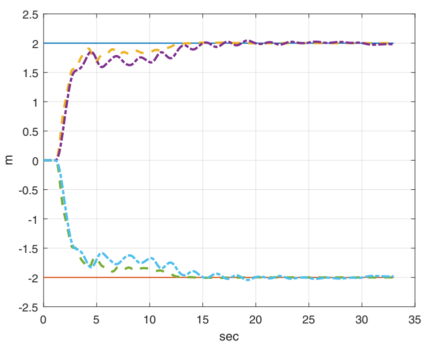
The result of the uplink delay compensation described in Section 3 is shown in Fig. 3, where we command the reference position (2,-2) to the two UAVs whose initial positions are (0,0). In this simulation, there is no downlink delay. As the delay increases from 0 to 1.5 sec, the settling time and the 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.

Fig. 4 summarizes the comparison between the linear compensation and the proposed GP-based compensation with respect to variation of the downlink delays. Figs. 4(a) to 4(f) are the results when constant references (2,-2) are given. The downlink delays are defined as Gaussian distributions with mean $\mu(\tau_d)$ equal to 0.1, 1.5, and 2.5 sec, while the variance is fixed at 0.2 sec. As the delay increases, the linear compensation lose the tracking accuracy as shown in Figs. 4(b) and 4(c). Particularly in Fig. 4(c), the linear compensation method manages to track the reference only to 20 sec. After that, one of the UAVs becomes unstable because it cannot overcome the time delay. On the other hand, because the GP method learns situations in regard to the delays, it becomes robust and maintains accurate performance as shown in Fig 4(f).

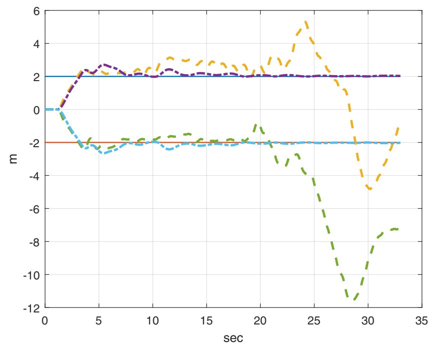
Figs. 4(g) to 4(i) illustrate another tracking scenario whose task is to follow rectangular waypoints in a finite period of time. In this scenario, the mean and the variance of the Gaussian delay are set to 2 and 0.2 sec, respectively. The figures show the trajectories of the first UAV and the comparison between the GP-based and the linear compensa-



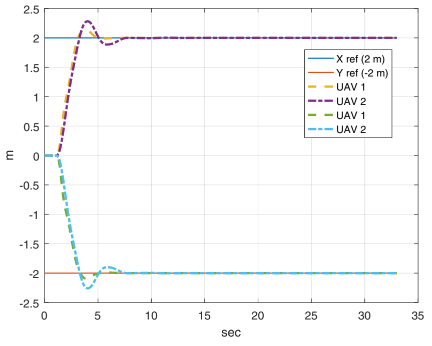
(a) Linear compensation, $\mu(\tau_d) = 0.1$ sec



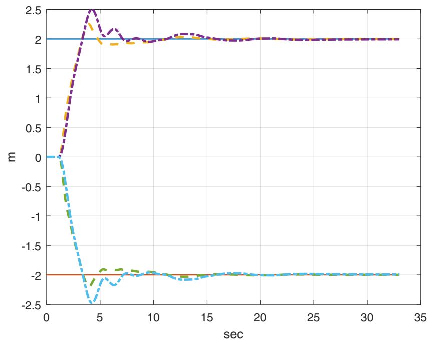
(b) Linear compensation, $\mu(\tau_d) = 1.5$ sec



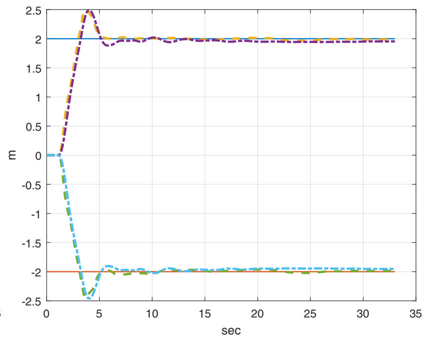
(c) Linear compensation, $\mu(\tau_d) = 2.5$ sec



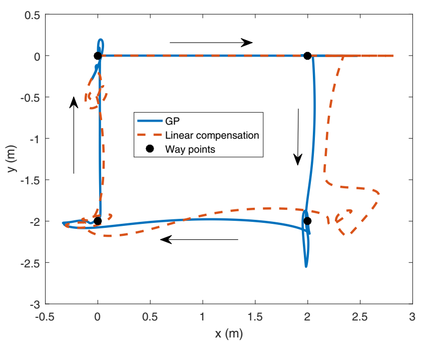
(d) GP, $\mu(\tau_d) = 0.1$ sec



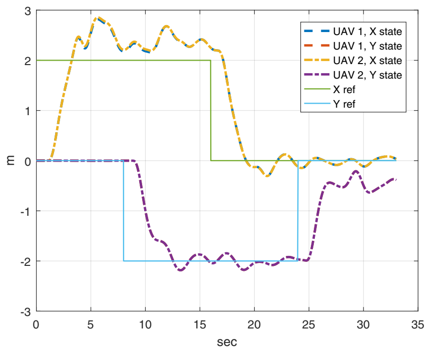
(e) GP, $\mu(\tau_d) = 1.5$ sec



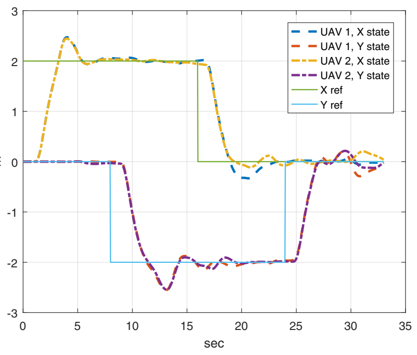
(f) GP, $\mu(\tau_d) = 2.5$ sec



(g) UAV 1 trajectory, $\mu(\tau_d) = 2.5$ sec



(h) Linear compensation, $\mu(\tau_d) = 2.5$ sec



(i) GP, $\mu(\tau_d) = 2.5$ sec

Fig. 4. Comparison between the linear compensation and the GP-based compensation with respect to a constant reference (a)–(f) and waypoints reference (g)–(i), with variations of the downlink delay.

tions. The UAV based on the linear compensation method cannot reach the reference and it wanders around the waypoints due to inaccurate delay compensation. Meanwhile, the GP-based control shows better tracking performance as shown in Fig. 4(i).

Let us discuss how the GP compensation improves the tracking results. According to equation (7), the GP-based machine learning method aims to learn the difference between the delayed observations and the true states. This can be interpreted as the GP-based compensation tries to imitate the situation where there is no delay. For example, the tracking result in Fig. 4(d) (almost no delay) illustrates the convergence after the short overshoot and undershoot. We also find similar responses from Figs. 4(e) and 4(f) in which the GP learning is used. Therefore, as long as the GP learns a model successfully, the resultant trajectory is similar to the zero-delay tracking result.

Finally, in order to analyze how many iterations of the training is necessary for guaranteeing accurate control performance, Fig. 5 shows the training error defined as mean absolute deviation (MAD), with respect to the number of iterations. One iteration lasts for 33 second and generates 131 training data points. We use the same simulation setup as in Fig. 4(f). Because the two UAVs have different tracking performances, each MAD reduces at different speeds, see Fig. 5. The MAD increases when the learning system receives new training data points that are far away from the domain of the existing training data set. After training those new data points, the error instantly decreases. Thus, the errors repeatedly increase and decrease to find a pattern inside the training data generated from the two UAVs maneuvers. We need about 20 iterations until convergence.

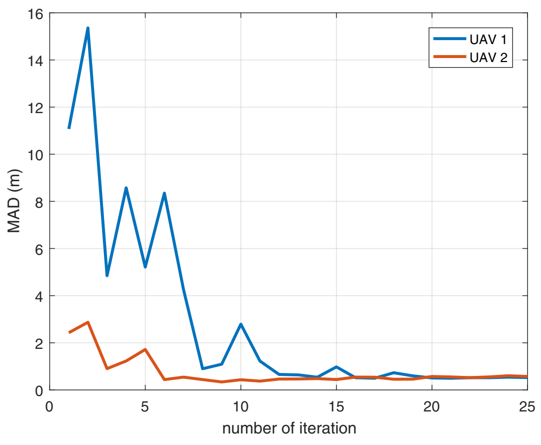


Fig. 5. Training error MAD with respect to the number of iterations.

6. CONCLUSION

We considered a scenario of collaborating UAVs with compensation of random communication time-delays. A GP-based machine learning approach is used to learn the network-induced effects, and the combination with a deterministic model enhances compensation capability. From simulation results, we see that the tracking results are improved thanks to the accurate learning. As future work, we will increase the number of UAVs and develop realistic field experiments.

REFERENCES

Akametalu, A.K., Fisac, J.F., Gillula, J.H., Kaynama, S., Zeilinger, M.N., and Tomlin, C.J. (2014). Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, 1424–1431. IEEE.

Berkenkamp, F., Schoellig, A.P., and Krause, A. (2015). Safe controller optimization for quadrotors with gaussian processes. *arXiv preprint arXiv:1509.01066*.

Chao, H., Baumann, M., Jensen, A., Chen, Y., Cao, Y., Ren, W., and McKee, M. (2008). Band-reconfigurable multi-uav-based cooperative remote sensing for real-time water management and distributed irrigation control. *IFAC Proceedings Volumes*, 41(2), 11744–11749.

Corke, P.I. (1996). A robotics toolbox for matlab. *IEEE Robotics and Automation Magazine*, 3(1), 24–32.

Dong, H., Wang, Z., and Gao, H. (2010). Robust filtering for a class of nonlinear networked systems with multiple stochastic communication delays and packet dropouts. *IEEE Transactions on Signal Processing*, 58(4), 1957–1966.

Hu, W., Liu, G., and Rees, D. (2007). Event-driven networked predictive control. *IEEE Transactions on Industrial Electronics*, 54(3), 1603–1613.

Jeong, S.C. and Park, P. (2005). Constrained mpc algorithm for uncertain time-varying systems with state-delay. *IEEE Transactions on Automatic Control*, 50(2), 257–263.

Ko, J., Klein, D.J., Fox, D., and Haehnel, D. (2007). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 742–747. IEEE.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9, 235–284.

Lee, D., Kim, H.J., and Sastry, S. (2009). Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of control, Automation and systems*, 7(3), 419–428.

Li, Y., Jha, D.K., Ray, A., and Wettergren, T.A. (2015). Feature level sensor fusion for target detection in dynamic environments. In *2015 American Control Conference*, 2433–2438. IEEE.

Liu, G., Mu, J., Rees, D., and Chai, S. (2006). Design and stability analysis of networked control systems with random communication time delay using the modified mpc. *International Journal of Control*, 79(4), 288–297.

Lv, F., Wen, C., Bao, Z., and Liu, M. (2016). Fault diagnosis based on deep learning. In *2016 American Control Conference*, 6851–6856. IEEE.

Onat, A., Naskali, T., Parlakay, E., and Mutluer, O. (2011). Control over imperfect networks: Model-based predictive networked control systems. *IEEE Transactions on Industrial Electronics*, 58(3), 905–913.

Osborne, M.A., Roberts, S.J., Rogers, A., Ramchurn, S.D., and Jennings, N.R. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, 109–120.

Quaritsch, M., Kruggl, K., Wischounig-Strucl, D., Bhattacharya, S., Shah, M., and Rinner, B. (2010). Networked uavs as aerial sensor network for disaster management applications. *Elektrotechnik und Informationstechnik*, 127(3), 56–63.

Rasmussen, C.E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, 63–71. Springer.

Wang, R., Liu, G.P., Wang, W., Rees, D., and Zhao, Y.B. (2010). Control for networked predictive control systems based on the switched lyapunov function method. *IEEE transactions on industrial electronics*, 57(10), 3565–3571.

Yoo, J.H., Kim, W., and Kim, H.J. (2011). Event-driven gaussian process for object localization in wireless sensor networks. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2790–2795. IEEE.

Zhang, H., Shi, Y., and Liu, M. (2013). H step tracking control for networked discrete-time nonlinear systems with integral and predictive actions. *IEEE Transactions on Industrial Informatics*, 9(1), 337–345.

Zhao, Y.B., Kim, J., and Liu, G.P. (2011). Error bounded sensing for packet-based networked control systems. *IEEE Transactions on Industrial Electronics*, 58(5), 1980–1989.

Zuo, T., Min, H., Zhang, T., and Zhang, X. (2016). The self-tuning networked control system with online delay prediction. *Transactions of the Institute of Measurement and Control*, doi:10.1177/0142331216636189.