



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사학위논문

**모델 예측 제어와 네트워크 지연 보상
기법을 이용한 무인기의 네트워크 제어**

**Networked operation of a UAV using learning-based
model predictive control and delay compensation**

2019 년 2 월

서울대학교 대학원

기계항공공학부

장 도 현

모델 예측 제어와 네트워크 지연 보상 기 법을 이용한 무인기의 네트워크 제어

Networked operation of a UAV using learning-based
model predictive control and delay compensation

지도교수 김 현 진

이 논문을 공학석사 학위논문으로 제출함

2019 년 1 월

서울대학교 대학원

기계항공공학부

장 도 현

장도현의 공학석사 학위논문을 인준함

2018 년 12 월

위원장

김 유 단

부위원장

김 현 진

위원

백 찬 구



**Networked operation of a UAV using learning-based
model predictive control and delay compensation**

A Thesis

by

DOHYUN JANG

Presented to the Faculty of the Graduate School of
Seoul National University
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mechanical & Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

February 2019

to my

FAMILY

with love

Abstract

Networked operation of a UAV using learning-based model predictive control and delay compensation

Dohyun Jang

Department of Mechanical & Aerospace Engineering

The Graduate School

Seoul National University

This study addresses an operation of unmanned aerial vehicles (UAVs) in a network environment where there is time-varying network delay. The network delay entails undesirable effects on the stability of the UAV control system due to delayed state feedback and outdated control input. Although several networked control algorithms have been proposed to deal with the network delay, most existing studies have assumed that the plant dynamics is known and simple, or the network delay is constant. These assumptions are improper to multirotor-type UAVs because of their nonlinearity and time-sensitive characteristics. To deal with these problems, we propose a networked control system using model predictive control (MPC) designed under the consideration of multirotor characteristics. We also apply a Gaussian process (GP) to learn an unknown nonlinear model, which increases accuracy of path planning and state estimation. Flight experiments show that the proposed algorithm successfully compensates the network delay and Gaussian process learning improves the UAV's path tracking performance.

Keyword : Networked control systems (NCS), Gaussian process (GP), Model predictive control (MPC), Delay compensation.

Student Number : 2017-21089

Table of Contents

	Page
Abstract	iii
Table of Contents	iv
List of Figures	v
List of Tables	vi
Chapter	
1 Introduction	1
1.1 Literature review	2
1.2 Thesis contribution	3
1.3 Thesis outline	3
2 Problem statement	4
2.1 GP-MPC for path planning	6
2.2 Uplink delay compensation	7
2.3 Downlink delay compensation	8
2.4 Clock synchronization	9
3 Model learning using Gaussian process	10
3.1 System dynamics for multirotor	10
3.2 Gaussian process to improve dynamic model	11
4 Model predictive control for networked UAV	14
4.1 MPC formulation	14
4.2 MPC formulation for networked control systems	15
5 Flight experiment	17
5.1 Delay analysis	17
5.2 Experimental setup	18
5.3 Experiment 1: circular flight with network delays	20
5.4 Experiment 2: two UAVs control with different network delays	24
6 Conclusion	27

List of Figures

1.1	Remote controlled UAVs via cloud network	2
2.1	The overall structure of the proposed algorithm: ① GP-MPC path planner (in server) sends control horizon $U(k)$, ② uplink delay compensator (in client) chooses proper control input \mathbf{u}_* in the delayed control horizon $U(k - \tau_u)$ according to the current time, ③ UAV inner control loop (in client) performs cascade control and sends full-state observation $\mathbf{x}(k)$ of the UAV, ④ downlink delay compensator (in server) estimates the current state of UAV $\hat{\mathbf{x}}(k)$ by compensating for downlink delay	5
2.2	Visual discription of uplink delay compensation	7
2.3	clock synchronization	9
5.1	Delay analysis between SNU (South Korea) and KTH (Sweden) (a) round-trip delay for 600 seconds (b) histogram of the round-trip delay	18
5.2	A snapshot of the experimental setup of the trajectory tracking control with NCS configuration	19
5.3	Experiment 1-(a): circular flight with network delays - without delay compensation	21
5.4	Experiment 1-(b): circular flight with network delays - with delay compensation, without GP learning	22
5.5	Experiment 1-(c): circular flight with network delays - with delay compensation, with GP learning	23
5.6	Experiment 2-(a): Two UAVs control with different network delays - without delay compensation	25
5.7	Experiment 2-(b): Two UAVs control with different network delays - with delay compensation, with GP learning	26

List of Tables

5.1 RMSE corresponding to given trajectory	24
--	----

1

Introduction

Unmanned aerial vehicles (UAVs) offer promising versatility and agility to achieve a wide range of missions at low costs [1–4]. Many UAV applications can show more outstanding effectiveness if the UAVs are controlled over longer distance [5]. However, long distance communication increases the time delay in a network, and the irregular, long time delay may degrade the control performance due to following reasons: 1) the observed UAV state in a remote side does not match the current UAV state due to the delayed state feedback, 2) the UAV also receives the outdated control input from the remote. Nevertheless, UAV research has often ignored the negative impact of the communication delay and most experiments were performed in well-equipped communication environments such as indoor laboratories.

Networked control system (NCS) approaches have been studied to overcome the problems related to the delay. NCS is a control framework to integrate many sensors, controllers, and plants at different geographical locations and to exchange signal over communication networks [6,7]. The most striking difference between the NCS and the other control systems is that the NCS uses a general-purpose network for various irrelevant yet concurrent applications [6], which means that the perfect communication is no longer assured.

Nonetheless, the NCS has some of the advantages: 1) it can address the network delay induced in a control loop, 2) various sensors, server, controller and plant can be connected simultaneously,

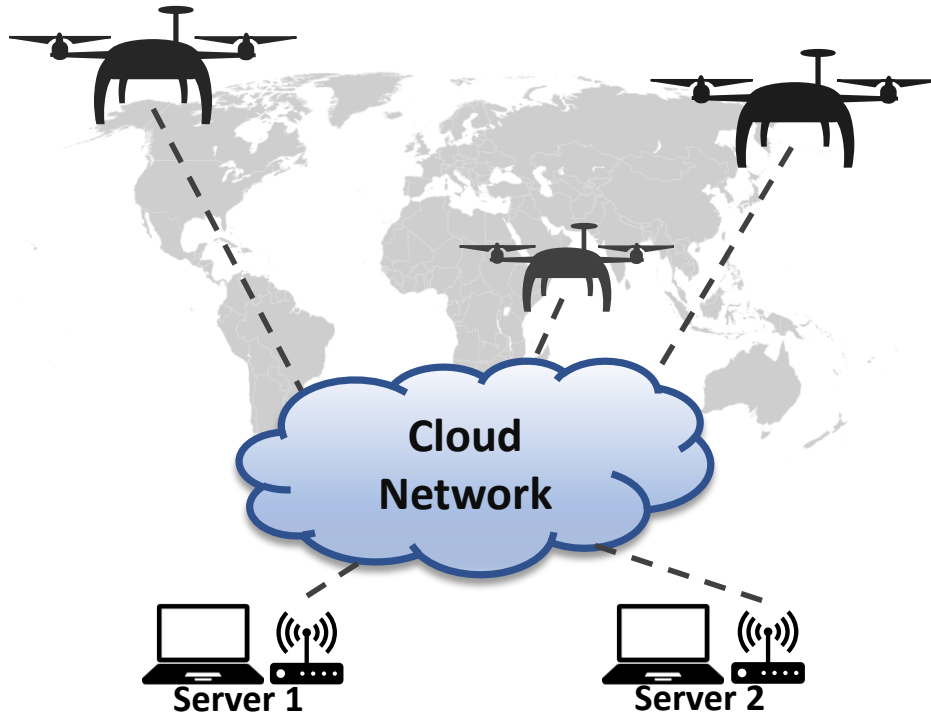


Figure 1.1: Remote controlled UAVs via cloud network

and 3) it is suitable for small plants such as the UAV because it requires less computing power, small memory space by performing complicated control and utilizing a large amount of database on the server side. This mechanism is called as local simple and remote complex (LSRC).

1.1 Literature review

There have been several researches that take advantage of the NCS in remote control. In [8–10], predictive control approaches are taken to provide a local plant with a sequence of predicted control inputs. Then, the local plant chooses a proper control input corresponding to the current network condition. These papers assume that the plant dynamics is known and simple such as a single servo motor. On the other hand, the multirotor dynamics that we are interested in is not so simple and may not be precisely known especially in the NCS setting. A small difference in the dynamics can cause an unexpected movement, or even crash in the worst case. In [11, 12], they tried to learn the network delay itself and used it for the networked control. However, both

only learn the approximate tendency of network delay, thus cannot cope with the volatile delay. In [13,14], the NCS problems for the UAV are addressed. However, both papers also assume that the plant dynamics is known, and network delays are time-invariant.

To solve problems in a more realistic environment, this study aims at establishing the NCS connecting SNU (South Korea) and KTH (Sweden) for a cooperative flight of the multirotor-type UAVs. We build an internet-based networked control system and design a path planning algorithm for the multirotor against a time-varying delay as shown in Fig. 1.1. Considering the characteristic of the NCS, we assume that the exact plant physical properties are unknown.

1.2 Thesis contribution

The main contribution of this paper is to design the networked controller using model predictive control (MPC) for the multirotor platforms. It can cope with not only the time-varying network delay but also any type of delay due to calculation or transmission. A machine learning technique is applied to improve the control performance by learning the multirotor dynamic models. It does not learn the time delay itself but learns a multirotor's unknown nonlinear model. Thus, we do not need a precise dynamic model in advance.

We employ the Gaussian process (GP) to learn the multirotor dynamics. The GP is an algorithm that has received much attention in recent years and has been widely used in applications including the UAV control [15] and the model learning of the MPC [16,17]. We utilize a sparse GP technique with the fully independent training conditional algorithm (FITC), which is less computationally intensive than the general GP algorithm.

1.3 Thesis outline

The remainder of this paper is organized as follows. In Section 2, we propose the problem statement which will be introduced. Section 3 explains the plant model learning with the GP. Section 4 introduces the design process of the MPC for networked UAV, and Section 5 details real-time experiment results. The final Section discusses the results and the control performance improvements.

2

Problem statement

We present the proposed NCS configuration for a multirotor system to compensate the time-varying network delay occurring in the networked control situation. The overall structure can be divided into two parts, one on the server side and the other on the client side. In this study, the remote side is the server, and the multirotor is the client. They communicate fixed-size data, called packets, for the control loop. The uplink delay τ_u occurs when the packets are transmitted from the server to the client. The downlink delay τ_d occurs in the opposite case. To configure the NCS, we suggest a compensation method that consists of the following four parts.

- Part 1: GP-MPC for path planning (Section 2.1)
- Part 2: uplink delay compensation (Section 2.2)
- Part 3: UAV inner control loop
- Part 4: downlink delay compensation (Section 2.3)

Fig. 2.1 shows overall structure. The MPC path planner in the server solves an optimization problem to predict a trajectory and results a control input set during H time steps. It puts the predicted control horizon in the packet with the timestamp and sends it to the client, during which the uplink time delay occurs. The uplink delay compensator calculates τ_u on the client

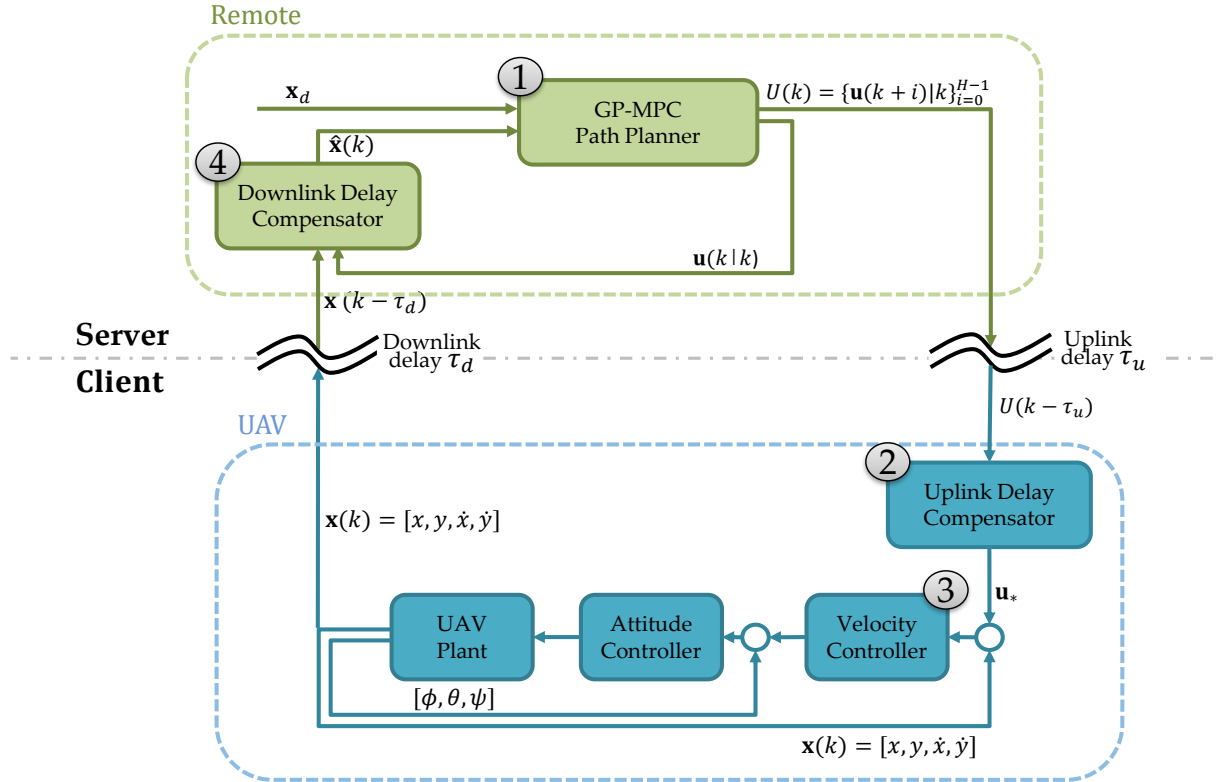


Figure 2.1: The overall structure of the proposed algorithm: ① GP-MPC path planner (in server) sends control horizon $U(k)$, ② uplink delay compensator (in client) chooses proper control input \mathbf{u}_* in the delayed control horizon $U(k - \tau_u)$ according to the current time, ③ UAV inner control loop (in client) performs cascade control and sends full-state observation $\mathbf{x}(k)$ of the UAV, ④ downlink delay compensator (in server) estimates the current state of UAV $\hat{\mathbf{x}}(k)$ by compensating for downlink delay

side by comparing the timestamp when the packet was generated and the current time and sends proper control input to the UAV according to τ_u . The UAV's inner controller conducts a cascade control with the received control input and sends out a full state feedback to the server. In the server, the downlink delay compensator calculates τ_d and estimates the current state of the UAV using a control input history. Estimated values are also used in the MPC path planner again.

2.1 GP-MPC for path planning

Let us define the state variables of the UAV as $\mathbf{x} := [\mathbf{p}^T \mathbf{v}^T]^T := [x \ y \ \dot{x} \ \dot{y}]^T \in \mathbb{R}^4$. It includes the position $\mathbf{p} \in \mathbb{R}^2$ and the velocities $\mathbf{v} \in \mathbb{R}^2$ in the inertial frame. The proposed setting works in the exactly same manner in 3-D, but we use 2-D notation for simplicity. $\mathbf{x}_d := [x_d \ y_d \ \dot{x}_d \ \dot{y}_d]^T \in \mathbb{R}^4$ is the desired position and velocities, $\mathbf{u} := [u_x \ u_y]^T \in \mathbb{R}^2$ is the control input vector. We assume that the altitude and yaw angle of the UAV are well controlled. The reason why the state variables do not include roll and pitch angle unlike the previous works [1] is because the attitude of the UAV changes rapidly, either estimating or measuring the current roll and pitch angle in the network environment with time delay is not reasonable.

The main objective of this research is to follow the desired trajectory with minimum deviation. We use the MPC for path planning, which tries to minimize the trajectory deviation $\sum_{i=1}^H \|\mathbf{x}(k+i) - \mathbf{x}_d(k+i)\| \rightarrow 0$ during the look-ahead horizon H . The multirotor dynamic model $f(\mathbf{x}, \mathbf{u})$ for the MPC is set as

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k)) \\ &= f_n(\mathbf{x}(k), \mathbf{u}(k)) + g(\mathbf{x}(k), \mathbf{u}(k)). \end{aligned} \quad (2.1)$$

$f_n(\mathbf{x}, \mathbf{u})$ is the nominal known model which is derived in (3.1)–(3.4), $g(\mathbf{x}, \mathbf{u})$ is an unknown nonlinear model, to be learned by the GP. Setting the multirotor dynamics as the sum of nominal and data-driven model via the GP improves the prediction accuracy because these models are complementary. In other words, the GP model can supplement the nominal model’s residual dynamics, and the nominal model can mitigate the failure of the GP prediction because the GP tends to result a zero output when the input of the sample is not around the domain of the existing training data set [5]. Using these definitions, we obtain optimized control horizon $U(k)$ and prediction horizon $X(k)$ as

$$\begin{aligned} U(k) &= \{\mathbf{u}(k+i|k)\}_{i=0}^{H-1} \in \mathbb{R}^{2 \times H}, \\ X(k) &= \{\mathbf{x}(k+i|k)\}_{i=1}^H \in \mathbb{R}^{4 \times H}. \end{aligned} \quad (2.2)$$

Typical MPC executes only the first input $\mathbf{u}(k|k)$. However, to make use of the network advantage of the transmitting data packets, which can include large data sets in a single fixed-size package, a set of serial control inputs are packed and transmitted through the network at time k [10].

2.2 Uplink delay compensation

The client receives the packet including the control horizon and the timestamp. However, due to the uplink delay τ_u , the client receives a packet at τ_u time later than the time it was created. τ_u can be calculated by comparing the timestamp included in the data packet with the time when the client receives the packet. During this delay, the UAV is following the previous trajectory so that the current UAV state is expected to be at the predicted position $\mathbf{x}(k|k - \tau_u) \in X(k - \tau_u)$. The uplink delay compensator chooses the proper control input \mathbf{u}_* in the delayed control horizon $U(k - \tau_u)$ according to the current time,

$$\mathbf{u}_* = \mathbf{u}(k|k - \tau_u) \in U(k - \tau_u). \quad (2.3)$$

\mathbf{u}_* is given to the UAV velocity controller. Since we assumed the irregular and time-varying delay, the delayed control horizon $U(k - \tau_u)$ may not arrive every time step. In this case, we use a most recent received $U(k - \tau_u)$ with the recalculated uplink delay τ_u . Fig. 2.3 presents uplink delay compensation process.

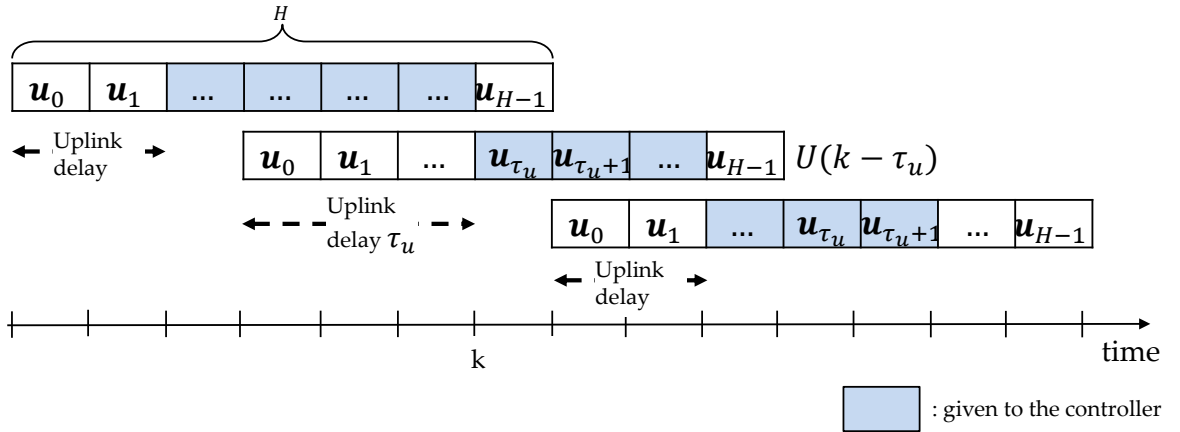


Figure 2.2: Visual discription of uplink delay compensation

2.3 Downlink delay compensation

The downlink delay τ_d can be calculated by comparing the timestamp included in the data packet with the time when the server receives the packet. First, we define the estimation function $f^{\{n\}}$ with a recurrence relationship using the dynamic model $f(\mathbf{x}, \mathbf{u})$. The server stores the history of past control inputs $\{\mathbf{u}(k-j|k-j)\}_{j=0}^{k-t_0}$, and the estimation function $f^{\{n\}}$ to calculate the estimated current position $\hat{\mathbf{x}}(k)$ is given by

$$\begin{aligned}
 f^{\{0\}}(\mathbf{x}(k)) &\triangleq \mathbf{x}(k) \\
 f^{\{1\}}(\mathbf{x}(k)) &\triangleq f(f^{\{0\}}(\mathbf{x}(k)), \mathbf{u}(k|k)) \\
 f^{\{2\}}(\mathbf{x}(k)) &\triangleq f(f^{\{1\}}(\mathbf{x}(k)), \mathbf{u}(k+1|k+1)) \\
 &\vdots \\
 f^{\{n+1\}}(\mathbf{x}(k)) &\triangleq f(f^{\{n\}}(\mathbf{x}(k)), \mathbf{u}(k+n|k+n)) \\
 &= \mathbf{x}(k+n+1), \quad (n = 0, 1, \dots)
 \end{aligned} \tag{2.4}$$

When the delayed observation $\hat{\mathbf{x}}(k - \tau_d)$ is given, $\hat{\mathbf{x}}(k)$ is calculated by the following equation:

$$\hat{\mathbf{x}}(k) = f^{\{\tau_d\}}(\hat{\mathbf{x}}(k - \tau_d)). \tag{2.5}$$

2.4 Clock synchronization

To use the packet-based delay compensation, clock synchronization should be performed. If the server and the client have a different clock, it is difficult to calculate the network delay τ_u and τ_d . To overcome the clock difference between the server and the client, we design a clock synchronizer. At first, the client sends a data packet containing a current timestamp $t_{client}(t_0)$ to the server. When the server receives this packet, the server sends again to the client with a current timestamp $t_{server}(t_1)$ measured by the server's clock. The client can calculate the round-trip delay t_{round} using the timestamp $t_{client}(t_0)$ and current time $t_{client}(t_2)$ as follows,

$$t_{round} = t_{client}(t_2) - t_{client}(t_0). \quad (2.6)$$

If we assume that the τ_u and τ_d are same at each time, which is the common and rational assumption, the one-way delay $t_{one-way}$ is a half of the t_{round} . The new client time at t_0 , $t'_{client}(t_0)$, is $t_{server}(t_1) - t_{one-way}$. We can calculate the time offset for the client as follows:

$$t_{offset,client} = t'_{client}(t_0) - t_{client}(t_0). \quad (2.7)$$

The time offset can be calculated in each client, should be added to the client timestamp.

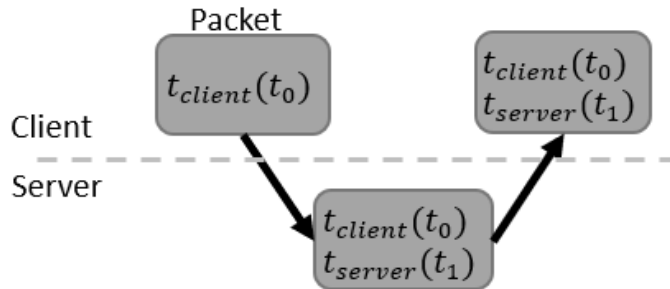


Figure 2.3: clock synchronization

3

Model learning using Gaussian process

In the previous section, we define the UAV's dynamic model $f(\mathbf{x}, \mathbf{u})$ as the sum of the nominal model $f_n(\mathbf{x}, \mathbf{u})$ and the unknown model $g(\mathbf{x}, \mathbf{u})$. We use a linear model of the UAV for the nominal model, which was derived from [18].

3.1 System dynamics for multirotor

We approximate the nominal model as a 1_{st} order dynamics and set the control input \mathbf{u} as a desired velocity of the UAV's velocity controller, which is more stable than setting the direct control input such as a desired moment or attitude because the velocity controller is less time-sensitive than both a motor thrust controller and attitude controller. If the UAV model can be considered as a point mass model, the dynamics equation of the UAV is defined as

$$\dot{\mathbf{x}} = A_c \mathbf{x} + B_c \mathbf{u}, \quad (3.1)$$

$$A_c \triangleq \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1/t_x & 0 \\ 0 & 0 & 0 & -1/t_y \end{bmatrix}, \quad (3.2)$$

$$B_c \triangleq \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/t_x & 0 \\ 0 & 1/t_y \end{bmatrix}. \quad (3.3)$$

Time constants t_x and t_y in the 1st order dynamics of \mathbf{v} can be determined experimentally [19]. Using (3.1), the difference equation for discrete system can be derived as

$$\begin{aligned} \mathbf{x}(k+1) &= A_d \mathbf{x}(k) + B_d \mathbf{u}(k) \\ &= f_n(\mathbf{x}(k), \mathbf{u}(k)), \end{aligned} \quad (3.4)$$

where A_d and B_d in the discrete domain correspond to A_c and B_c in the continuous version, respectively.

3.2 Gaussian process to improve dynamic model

However, using an approximated dynamic model can cause inaccurate prediction horizon $X(k)$. Erroneous prediction horizon is more harmful in the NCS because both uplink and downlink delay compensators assume that the UAV follows the prediction horizon. To generate more accurate prediction horizon, the GP learns the unknown UAV model $g(\mathbf{x}, \mathbf{u})$ with state \mathbf{x} and control input \mathbf{u} . The main downside of the GP is a computational burden. To overcome this drawback, the sparse GP was developed. The sparse GP makes it possible to reduce the runtime by making assumptions about a prior distribution. In this paper, we use the sparse GP with the fully independent training conditional algorithm (FITC), which is introduced in [20].

To learn such an unknown model $g(\mathbf{x}, \mathbf{u})$ using the GP modelling techniques, we define the state control tuple $\tilde{\mathbf{x}}(k)$ and the residual model $\mathbf{z}(k)$ as follows:

$$\begin{aligned}
\tilde{\mathbf{x}}(k) &= [\mathbf{v}(k)^T \mathbf{u}(k)^T]^T \in \mathbb{R}^4, \\
\mathbf{z}(k) &= g(\mathbf{x}(k), \mathbf{u}(k)) + \varepsilon \\
&= \mathbf{x}(k+1) - f_n(\mathbf{x}(k), \mathbf{u}(k)) + \varepsilon \in \mathbb{R}^4.
\end{aligned} \tag{3.5}$$

$\tilde{\mathbf{x}}(k)$ does not include $\mathbf{p}(k)$ because it does not affect the residual model $\mathbf{z}(k)$. We assume that the output of the function $g(\mathbf{x}(k), \mathbf{u}(k))$ is corrupted by white noise ε with variance σ_n . $\tilde{\mathbf{x}}_*$ is a GP test input data and \mathbf{z}_* is a GP test target data. To predict the GP test target data, we first acquire the GP training input data $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}(i)\}_{i=1}^N$ and the training target data $\mathbf{Z} = \{\mathbf{z}(i)\}_{i=1}^N$. Then, we assume that the prior distribution of \mathbf{Z} and \mathbf{z}_* have a joint Gaussian distribution with zero-mean written as

$$\begin{aligned}
\begin{bmatrix} \mathbf{Z} \\ \mathbf{z}_* \end{bmatrix} &\sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} k(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n I & k(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*) \\ k(\tilde{\mathbf{x}}_*, \tilde{\mathbf{X}}) & k(\tilde{\mathbf{x}}_*, \tilde{\mathbf{x}}_*) \end{bmatrix} \right) \\
&= \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} K_{zz} + \sigma_n I & K_{z*} \\ K_{*z} & K_{**} \end{bmatrix} \right),
\end{aligned} \tag{3.6}$$

In this paper, the squared-exponential kernel function k is used, which is defined as

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}^*) = \sigma_s^2 \left(-\frac{1}{2} (\tilde{\mathbf{x}} - \tilde{\mathbf{x}}^*)^T \Sigma^{-1} (\tilde{\mathbf{x}} - \tilde{\mathbf{x}}^*) \right), \tag{3.7}$$

where σ_s^2 is the variance of the function $g(\mathbf{x}, \mathbf{u})$ and Σ is the length scale that determines how fast the correlation between data points decreases. The hyper parameters represent the smoothness of the function estimated by the GP. Typically, the hyper parameters can be learned by evidence maximization [21]. The posteriori distribution of \mathbf{z}_* is derived as follows:

$$\begin{aligned}
p(\mathbf{z}_* | \tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*) &\sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \\
\boldsymbol{\mu}_* &= K_{*z} (K_{zz} + \sigma_n I)^{-1} \mathbf{Z} \\
&= g(\mathbf{x}, \mathbf{u}) \\
\boldsymbol{\Sigma}_* &= K_{**} - K_{*z} (K_{zz} + \sigma_n I)^{-1} K_{z*}
\end{aligned} \tag{3.8}$$

We use the sparse GP to reduce the computational complexity. The sparse GP starts with the generation of the inducing input data $\tilde{\mathbf{x}}_c$ and corresponding target data \mathbf{z}_c . With the sparse GP algorithm and inducing GP data, the probability of \mathbf{z}_* can be calculated with a lower computational cost compared with that of nominal GP model [22].

As a result, we learn the unknown model $g(\mathbf{x}, \mathbf{u})$ and obtain the total dynamic equation $f(\mathbf{x}, \mathbf{u})$. It is used for MPC path planning in section 2.1, and downlink delay compensation in section 2.3.

4

Model predictive control for networked UAV

We apply model predictive control (MPC) for the multirotor platforms in networked control systems. It can cope with the time-varying network delay. The GP model obtained previous chapter is applied to improve the control performance by learning the multirotor dynamic models.

4.1 MPC formulation

In the previous chapter, we introduced the nominal model $f_n(\mathbf{x}, \mathbf{u})$ and the GP model $g(\mathbf{x}, \mathbf{u})$. We use a total dynamic model of the UAV $f(\mathbf{x}, \mathbf{u})$ as the model constraint in the following MPC setup:

$$\begin{aligned} \min_{\mathbf{u}(k+i|k), 0 \leq i < H} J_k = & \|\mathbf{x}(k+H) - \mathbf{x}_d(k+H)\|_P^2 \\ & + \sum_{i=0}^{H-1} (\|\mathbf{x}(k+i) - \mathbf{x}_d(k+i)\|_Q^2 + \|\mathbf{u}(k+i|k)\|_R^2) \end{aligned} \quad (4.1)$$

subject to

$$\begin{aligned}
\mathbf{x}(k+i+1) &= f(\mathbf{x}(k+i), \mathbf{u}(k+i|k)) \\
|\mathbf{u}(k+i|k)| &\leq \mathbf{u}_{max} \\
i &= 0, \dots, H-1 \\
\hat{\mathbf{x}}(k) &= f^{\{\tau_d\}}(\hat{\mathbf{x}}(k-\tau_d)) \\
\mathbf{x}(k) &= \hat{\mathbf{x}}(k).
\end{aligned} \tag{4.2}$$

Here $\|\mathbf{k}\|_A^2$ is a quadratic form of vector \mathbf{k} with a positive semi-definite weighting matrix A . J_k is the cost function and $\hat{\mathbf{x}}(k)$ is the estimated state obtained in the downlink delay compensator described in section 2.3. The MPC calculates the state transition up to the look-ahead horizon of H steps. The positive semi-definite matrices P , Q , and R are weights for the final state error, i_{th} state error, and control input, respectively. The vector \mathbf{u}_{max} denotes the constraint of control input. The results of the MPC are signals defined in (2.2).

To solve for the optimal control problem in real time, a stable and fast optimal control solver is required. We use a Sequential Linear Quadratic (SLQ) solver whose speed and performance were previously demonstrated in agile flight experiments [23]. The detailed algorithm of SLQ for the MPC is shown in [24].

4.2 MPC formulation for networked control systems

In section 2.3, the downlink delay compensator uses $\mathbf{u}(k-i|k-i)$ to transfer $\mathbf{x}(k-i)$ to the next step $\mathbf{x}(k-i+1)$ in each $k-i$ time step. However, $\mathbf{u}(k-i|k-i-\tau_u) \in U(k-i-\tau_u)$ was used as an output of the uplink delay compensator, and the downlink delay compensator does not know τ_u because it varies every moment. The best strategy is to make $\mathbf{u}(k+i|k)$ and $\mathbf{u}(k+i|k-1)$ as close as possible. As a result, the server can choose the $(k-i)$ -th step control input with minimum difference from the actual control input used. To consider this changes to the MPC constraints, we redefine the cost function of (4.1) as follows:

$$\begin{aligned}
\min_{\mathbf{u}(k+i|k), 0 \leq i < H} J_k &= \|\mathbf{x}(k+H) - \mathbf{x}_d(k+H)\|_P^2 \\
&+ \sum_{i=0}^{H-1} (\|\mathbf{x}(k+i) - \mathbf{x}_d(k+i)\|_Q^2 + \|\mathbf{u}(k+i|k)\|_R^2) \\
&+ \sum_{i=0}^{H-2} (\|\mathbf{u}(k+i|k) - \mathbf{u}(k+i|k-1)\|_S^2)
\end{aligned} \tag{4.3}$$

Even though the inclusion of the last term decreases the optimal performance, it increases stability which is a more critical factor in networked control systems.

5

Flight experiment

The proposed algorithms are validated with the micro multirotor through experiments. At first, we briefly introduce the result of delay analysis in case of our experimental setup. Then, we present the experimental setup and results of tracking control experiments with a single multirotor and two multirotors. The discussion of results follows.

5.1 Delay analysis

This subsection shows the result of the network delay characteristics analysis between Korea and Sweden. It was confirmed that the network communication between Korea and Sweden is on average 20 nodes. The delays that occur at each node can be seen to follow the Poisson distribution. Thus, we can assume that the round-trip network delay which is a sum of each node delay follows the normal distribution according to the central limit theorem even though they have different parameters. Fig. 5.1 supports that our assumptions. For a fair comparison with each experiment, we set the artificial delays τ_u and τ_d to follow the normal distribution $\mathcal{N}(0.5, 0.1)$ with a random seed.

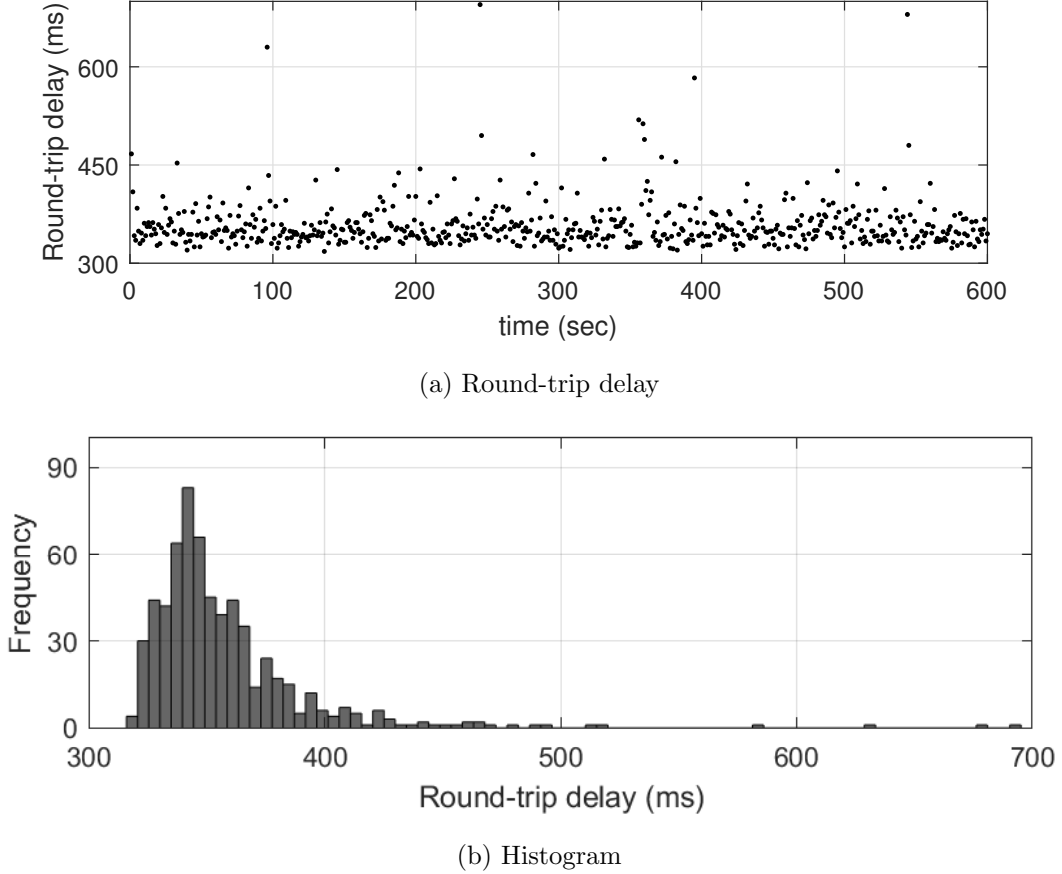


Figure 5.1: Delay analysis between SNU (South Korea) and KTH (Sweden) (a) round-trip delay for 600 seconds (b) histogram of the round-trip delay

5.2 Experimental setup

We validated the effectiveness of the proposed framework through actual flight experiments. The experimental setup is illustrated in Fig. 5.2. We use the Crazyflie 2.0 multirotor developed by Bitcraze [25], whose weight is approximately 32 g and maximum takeoff weight is 42 g. Position of the multirotor is measured by a VICON motion capture system operating at 100 Hz and all the other states are estimated using Kalman filter from the position information. We use a server computer with Intel i7 6700K 4.0GHz CPU and Robot Operating System (ROS). The server computer solves the MPC and sends the predicted control horizon to the multirotor. The MPC look-ahead horizon H is set to 20, and the MPC sampling time is 0.1 seconds. The constraint on

the control input is given by $\mathbf{u}_{max} = [0.6 \ 0.6]^T$.

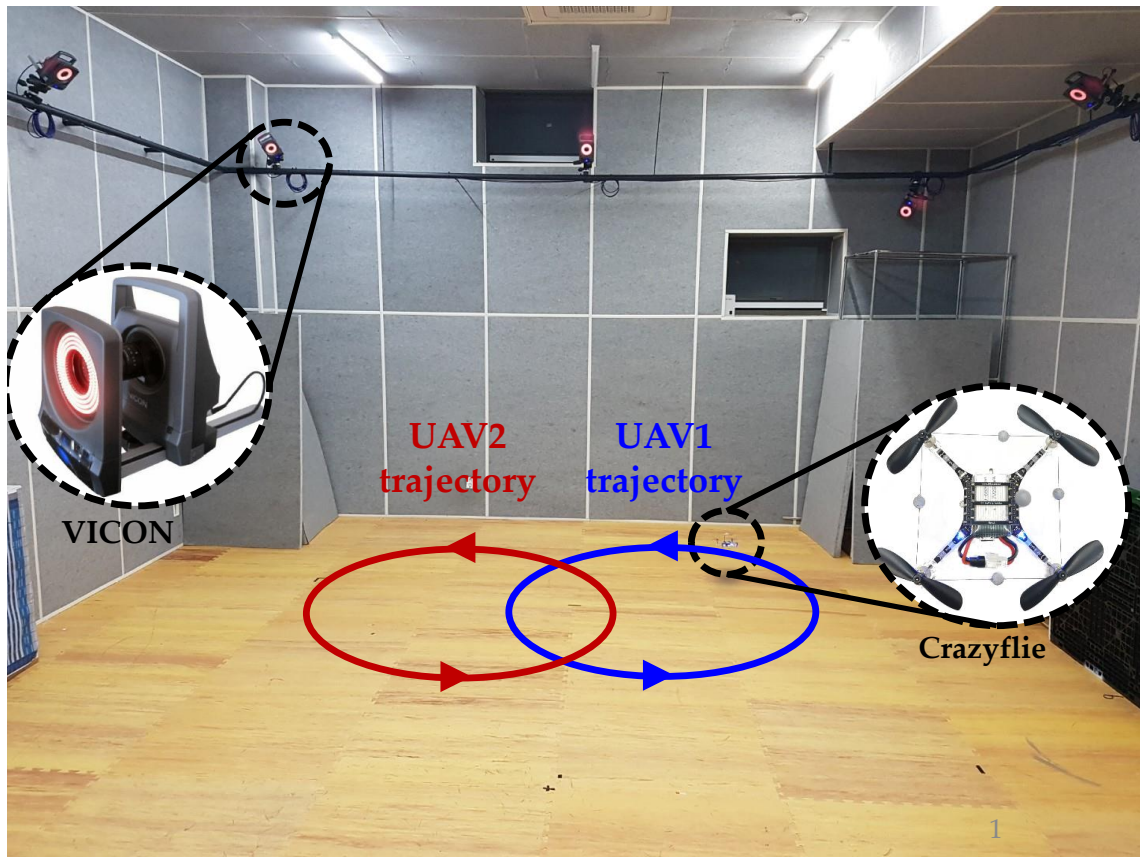


Figure 5.2: A snapshot of the experimental setup of the trajectory tracking control with NCS configuration

5.3 Experiment 1: circular flight with network delays

In this experiment, we conducted the path tracking control experiment in the presence of both uplink delay and downlink delay. We give a circular desired path with period 10 seconds, radius 0.5 meters.

Figs. 5.3 to 5.5 show the results of our first experiment, which compares three cases: (a) tracking control without any delay compensation, (b) the proposed delay compensation algorithm without the GP model learning, (c) the proposed delay compensation algorithm with the GP model learning. Even though the prior information of the time delay is not given to the delay compensator, tracking control performances of (b) and (c) are satisfactory because of the robustness of the proposed algorithm. Especially the learned GP model enhances the control performance because of more accurate prediction horizon and downlink delay compensation. Table 5.1 shows the comparison of tracking performance.

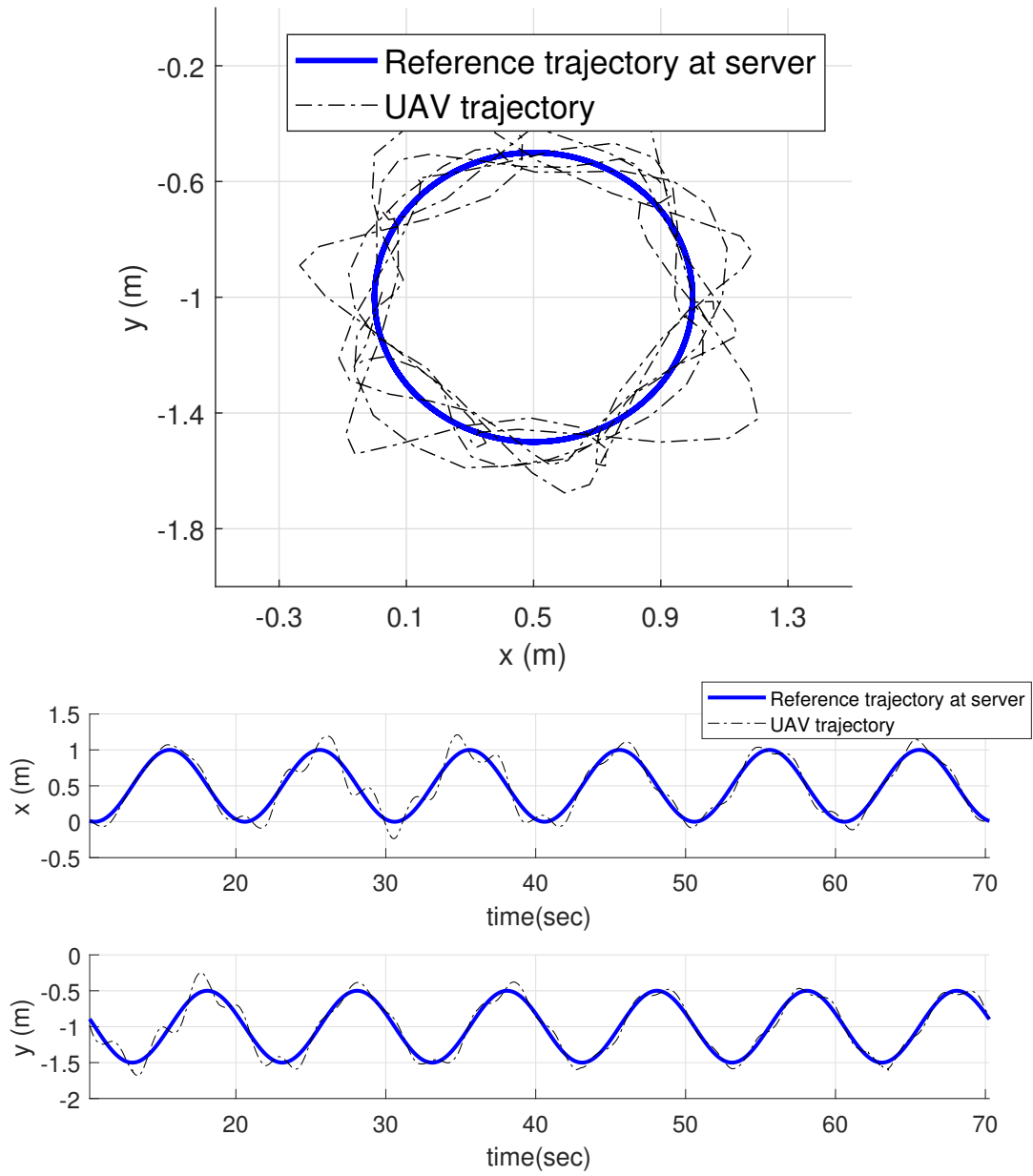


Figure 5.3: Experiment 1-(a): circular flight with network delays - without delay compensation

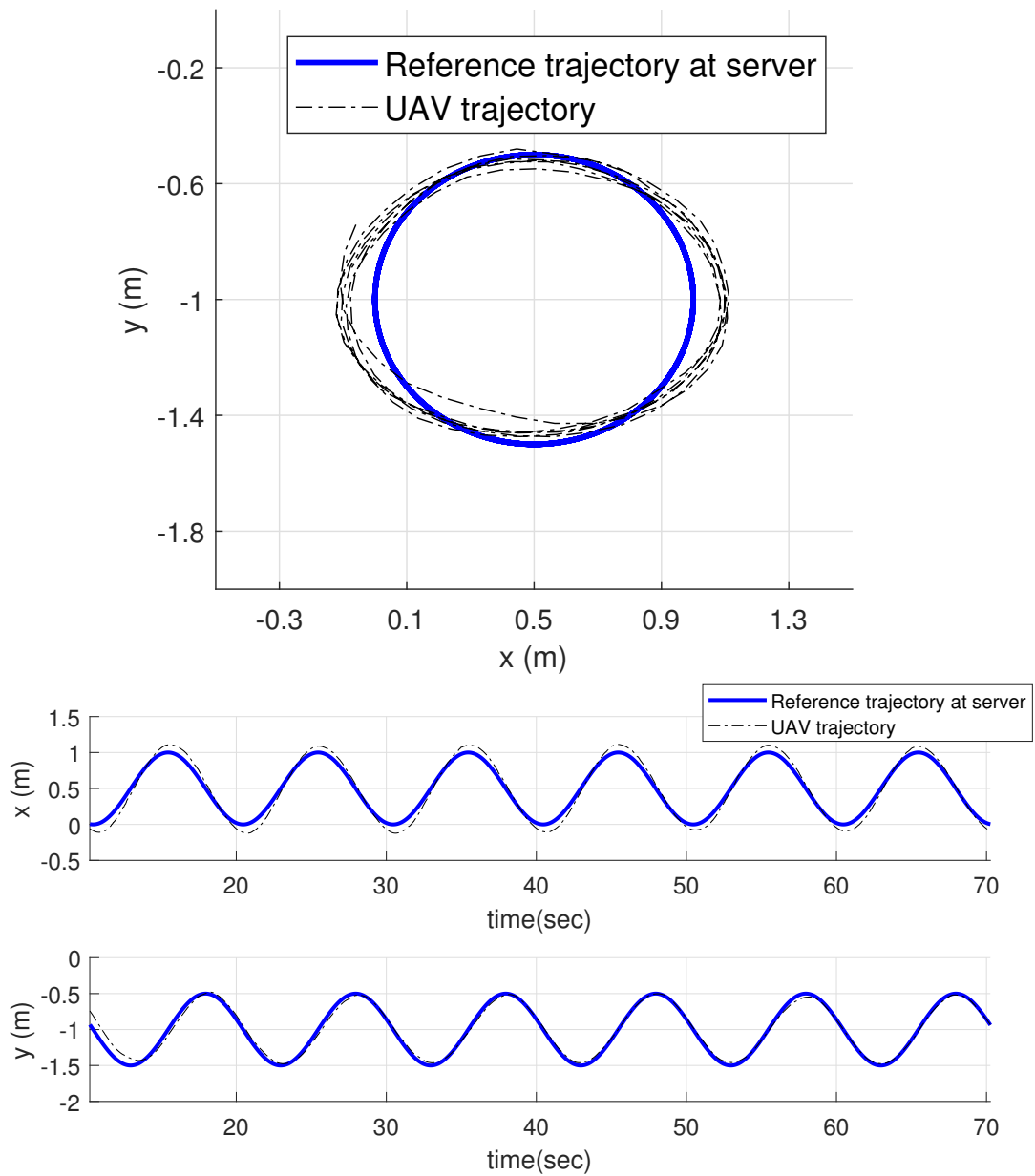


Figure 5.4: Experiment 1-(b): circular flight with network delays - with delay compensation, without GP learning

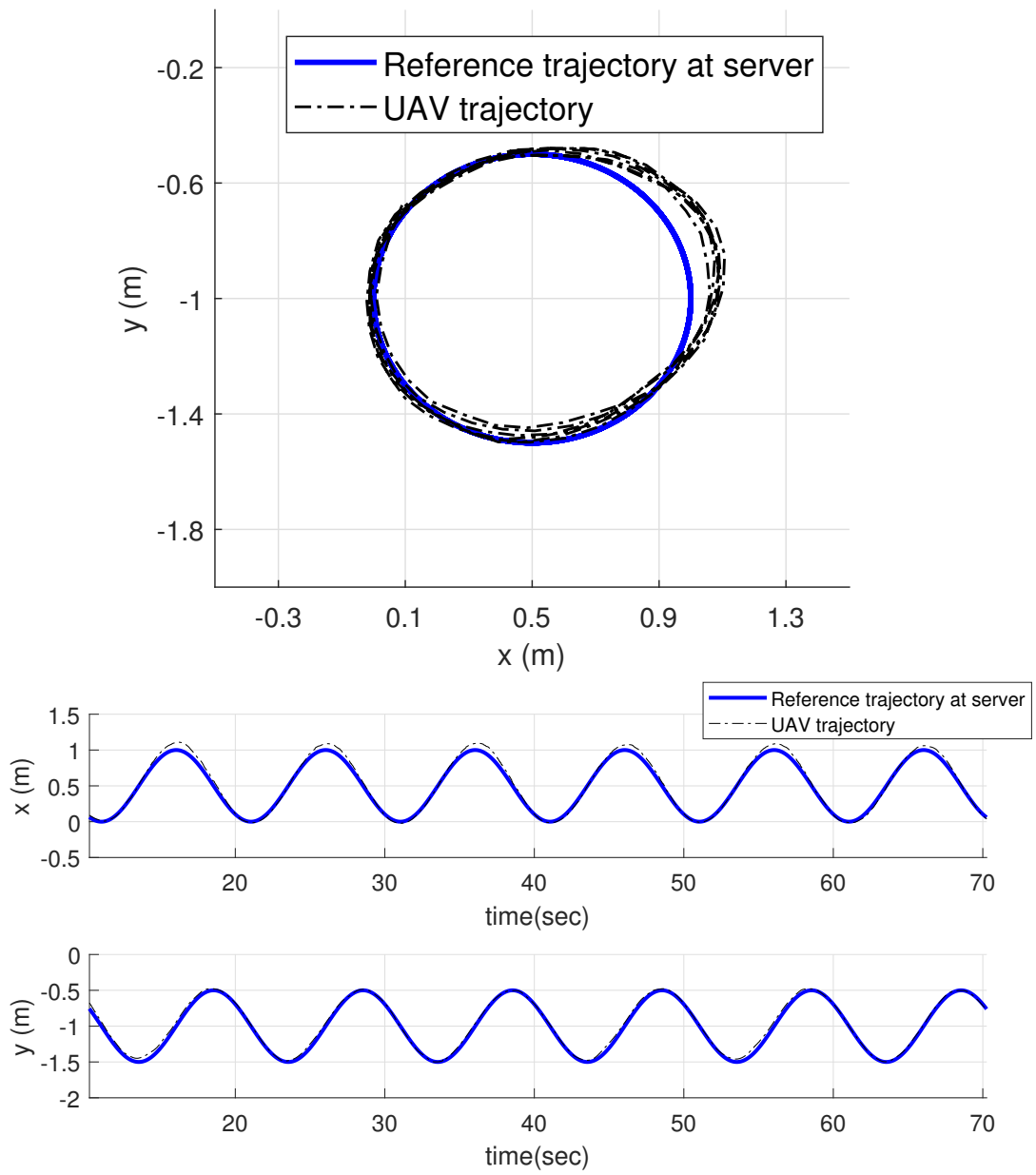


Figure 5.5: Experiment 1-(c): circular flight with network delays - with delay compensation, with GP learning

Table 5.1: RMSE corresponding to given trajectory

	Exp 1-(a)	Exp 1-(b)	Exp 1-(c)
X-RMSE	0.1048	0.0807	0.0507
Y-RMSE	0.0861	0.0495	0.0350
XY-RMSE	0.1352	0.0946	0.0616

5.4 Experiment 2: two UAVs control with different network delays

In this experiment, we conducted the cooperative path tracking control with two multirotors. We assume a scenario where each multirotor is manipulated in different locations. Thus, we set the both artificial delays τ_u and τ_d to follow the normal distribution $\mathcal{N}(0.5, 0.1)$ and $\mathcal{N}(0.3, 0.05)$ for each multirotor. We give a circular desired path with period 10 seconds, radius 0.5 meters. The two multirotors fly a half meter apart in x -axis. Figs. 5.6 and 5.7 show the results of our second experiment, which compares two cases: (a) cooperative tracking control without any delay compensation, (b) cooperative tracking control with the proposed delay compensation algorithm and the GP model learning. Even though the network delays of each multirotor are different, the proposed algorithm maintained the given distance compared to the result without compensation algorithm.

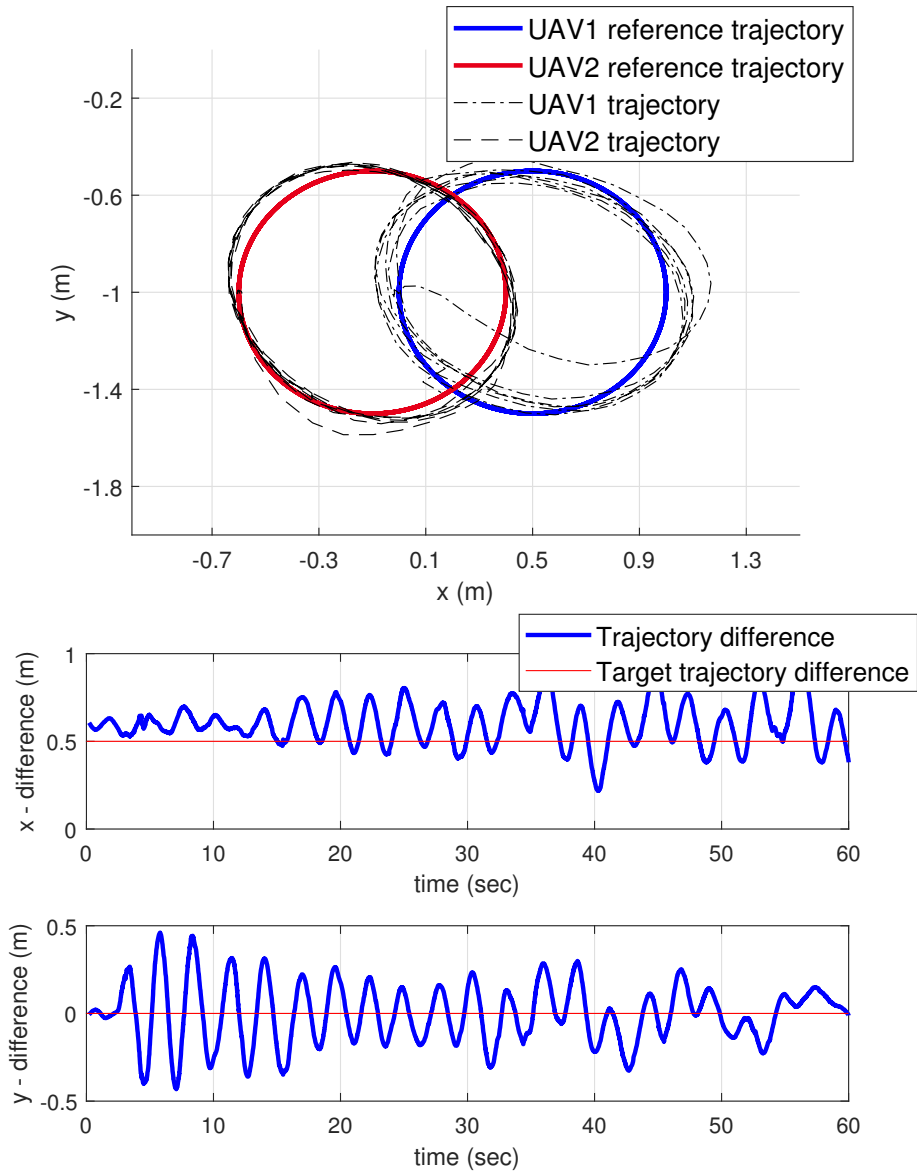


Figure 5.6: Experiment 2-(a): Two UAVs control with different network delays - without delay compensation

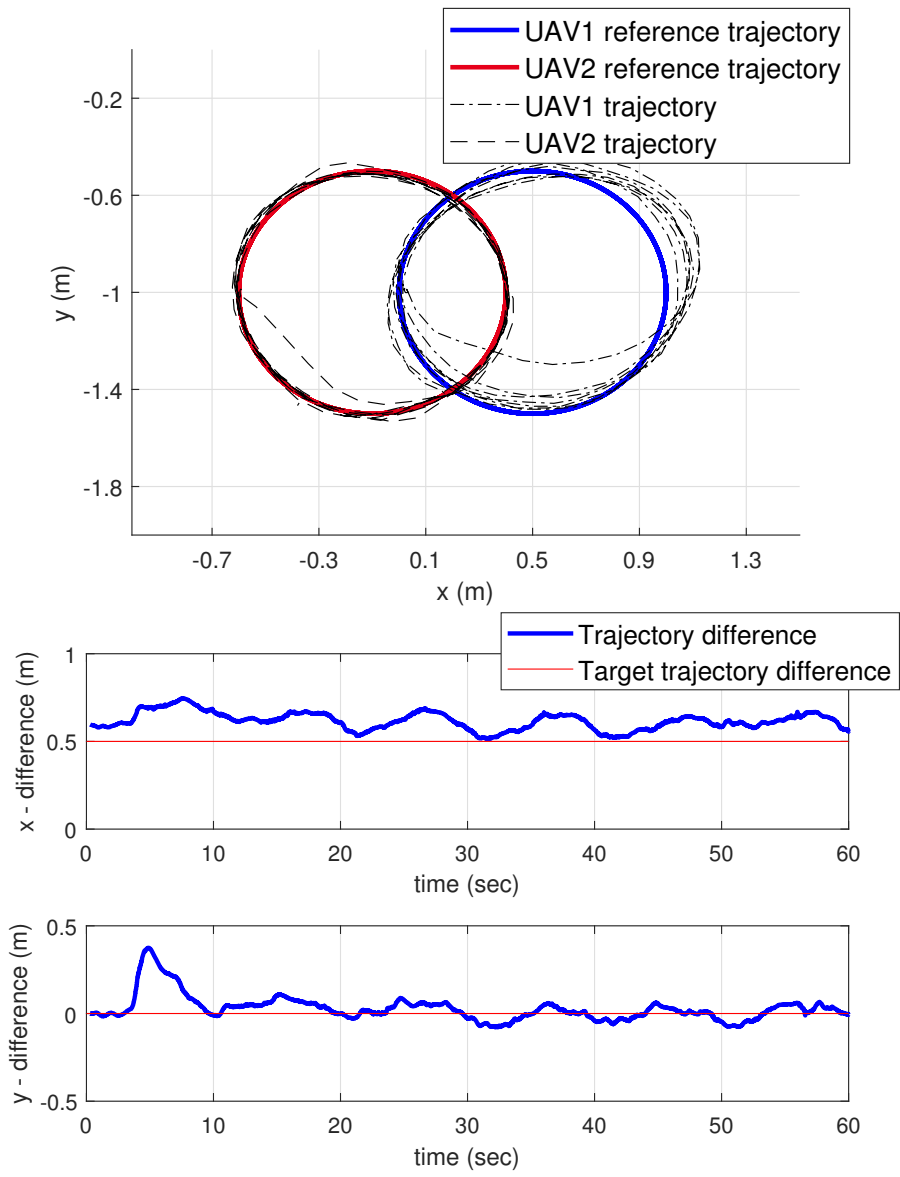


Figure 5.7: Experiment 2-(b): Two UAVs control with different network delays - with delay compensation, with GP learning

6

Conclusion

This paper presented a configuration of NCS for the UAV to compensate the time-varying network delay. The NCS structure proposed in this paper consists of 4 parts: GP-MPC path planner, uplink delay compensator, UAV inner-loop controller, and downlink delay compensator. This structure was configured to compensate the network delay and control the multirotor efficiently. We also used the GP model learning to improve the delay compensation performance and MPC path planning accuracy. The first experiment on tracking control of the multirotor UAV confirms that the control performance was improved. The second experiment on cooperative path tracking control of the two multirotors shows that the proposed algorithm can cope with unknown network delays.

References

- [1] S. Bouabdallah, P. Murrieri, and R. Siegwart, “Design and control of an indoor micro quadrotor,” in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4393–4398.
- [2] H. Lee, H. Kim, W. Kim, and H. J. Kim, “An integrated framework for cooperative aerial manipulators in unknown environments,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2307–2314, 2018.
- [3] M. Bernard, K. Kondak, I. Maza, and A. Ollero, “Autonomous transportation and deployment with aerial robots for search and rescue missions,” *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.
- [4] S. Kim, S. Choi, and H. J. Kim, “Aerial manipulation using a quadrotor with a two dof robotic arm,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4990–4995.
- [5] J. Yoo and K. H. Johansson, “Learning communication delay patterns for remotely controlled uav networks,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 216–13 221, 2017.
- [6] F.-Y. Wang and D. Liu, “Networked control systems,” *Theory and Applications, Springer-Verlag, London*, 2008.
- [7] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, “A survey of recent results in networked control systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.
- [8] L. A. Montestruque and P. J. Antsaklis, “On the model-based control of networked systems,” *Automatica*, vol. 39, no. 10, pp. 1837–1843, 2003.
- [9] P. L. Tang and C. W. de Silva, “Compensation for transmission delays in an ethernet-based control network using variable-horizon predictive control,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 707–718, 2006.

- [10] G.-P. Liu, “Analysis and design of networked predictive control systems,” in *Networked Control Systems*. Springer, 2008, pp. 95–119.
- [11] Y. Tipsuwan and M.-Y. Chow, “Control methodologies in networked control systems,” *Control engineering practice*, vol. 11, no. 10, pp. 1099–1111, 2003.
- [12] 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*, vol. 39, no. 9, pp. 1365–1373, 2017.
- [13] J. Yoo, S. Lee, H. J. Kim, and K. H. Johansson, “Trajectory generation for networked uavs using online learning for delay compensation,” in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, 2017, pp. 1941–1946.
- [14] J. Yoo, H. J. Kim, and K. H. Johansson, “Path planning for remotely controlled uavs using gaussian process filter,” in *Control, Automation and Systems (ICCAS), 2017 17th International Conference on*. IEEE, 2017, pp. 477–482.
- [15] F. Berkenkamp and A. P. Schoellig, “Safe and robust learning control with gaussian processes,” in *Control Conference (ECC), 2015 European*. IEEE, 2015, pp. 2496–2501.
- [16] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, “Gaussian process model based predictive control,” in *American Control Conference, 2004. Proceedings of the 2004*, vol. 3. IEEE, 2004, pp. 2214–2219.
- [17] G. Cao, E. M.-K. Lai, and F. Alam, “Gaussian process model predictive control of an unmanned quadrotor,” *Journal of Intelligent & Robotic Systems*, vol. 88, no. 1, pp. 147–162, 2017.
- [18] D. Jang, J. Yoo, , and H. J. Kim, “Tracking control of a multirotor uav in a network environment with time-varying delay.”
- [19] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, “Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system,” in *Robot Operating System (ROS)*. Springer, 2017, pp. 3–39.

- [20] J. Quiñero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate gaussian process regression,” *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.
- [21] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.
- [22] H. Bijl, T. B. Schön, J.-W. van Wingerden, and M. Verhaegen, “Online sparse gaussian process training with input noise,” *stat*, vol. 1050, p. 29, 2016.
- [23] D. W. Mellinger, “Trajectory generation and control for quadrotors,” 2012.
- [24] C. Y. Son, H. Seo, T. Kim, and H. J. Kim, “Model predictive control of a multi-rotor with a suspended load for avoiding obstacles.”
- [25] <https://www.bitcraze.io/>.

국 문 초 록

본 연구는 시간에 따라 변화하는 네트워크 지연이 존재하는 네트워크 환경에서의 무인 항공기 (UAV)의 제어 기법에 대하여 소개한다. 네트워크 지연은 주로 상태 피드백과 제어 입력의 지연을 야기시키고, 이로 인해 UAV 제어 시스템의 안정성에 악영향을 미친다. 이와 같은 네트워크 지연에 대처하기 위하여 몇 가지 네트워크 제어 알고리즘이 제안되었지만 대부분의 기존 연구에서는 플랜트 동역학이 매우 단순하거나 정확히 알고 있다고 가정하였고, 일정한 네트워크 지연이 발생하는 상황에서만 수행되었다. 하지만 이러한 가정은 비선형 모델 및 시간에 민감한 제어 특성을 가지는 멀티로터 형태의 UAV에 적합하지 않다. 이러한 문제를 해결하기 위하여 멀티로터의 특성을 고려하여 설계된 모델 예측 제어 (MPC)를 이용한 네트워크 제어 시스템을 제안한다. 또한 경로 계획 및 상태 추정의 정확도를 높이고자 가우시안 프로세스 (GP) 기법을 적용하여, 멀티로터 동역학에 고려되지 않은 미지의 모델을 학습하도록 한다. 실내 비행 실험을 통하여 제안된 알고리즘이 네트워크 지연을 효과적으로 보상하고 가우시안 프로세스 학습이 UAV의 경로 추적 성능을 향상시킨다는 것을 보여준다.

주요어 : 네트워크 제어 시스템, 가우시안 프로세스, 모델 예측 제어, 네트워크 지연 보상.

학번 : 2017-21089