Unmanned Systems, Vol. 6, No. 1 (2018) 39–65

DOI: 10.1142/S2301385018500036

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

High-Precision Multi-UAV Teaming for the First Outdoor Night Show in Singapore

Kevin Z. Y. Ang*,¶, Xiangxu Dong*, Wenqi Liu*, Geng Qin*, Shupeng Lai†, Kangli Wang†, Dong Wei‡, Songyuan Zhang‡, Phang Swee King*,§, Xudong Chen*, Mingjie Lao*, Zhaolin Yang*, Dandan Jia*, Feng Lin*, Lihua Xie‡, Ben M. Chen†

*Temasek Laboratories,
The National University of Singapore, Singapore 117411
†Department of Electrical and Computer Engineering,
The National University of Singapore, Singapore 117583
‡School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore 639798
§School of Engineering, Taylor's University, 1 Jalan Taylor's
47500 Subang Jaya, Selangor DE, Malaysia

Advancement in the development of automation in aerial robotics has created endless applications today by utilizing autonomous drones, or in other words, unmanned aerial vehicles (UAVs). Motivated by the idea of the entertainment robots, in this paper, we present a robust and safe multi-UAV formation system for the purpose of entertainment in the form of a night multi-UAV teaming light show. The performance includes a 5-minutes flight show with 16 UAVs, changing patterns with background music, and with synchronized visual lighting from each of the UAVs. The high-precision autonomous formation flight is achieved with a centralized control method with a single ground control station (GCS). The system includes offline trajectory generation, safety features such as collision avoidance and UAV states monitoring, geo-fencing for out-of-bound control and many more which will be discussed in this paper. The live performance of the multi-UAV teaming night show was successfully presented to the public in conjunction with The Future of Us Exhibitions held in January 2016, in Singapore.

Keywords: UAV; swarm; formation; autonomous.

US

1. Introduction

With the recent developments in sensor, communication and computational technology, various components which are necessary for on-board computer system construction have become increasingly smaller and more powerful than before. This led to the sudden increase in developmental and research work with unmanned aerial vehicles (UAVs). They have been successfully implemented in various flight missions such as reconnaissance in hostile territories. Among various UAV research directions, one of the critical research areas of UAVs is autonomous formation flight [1, 2]. Formation flight is defined as the collective use of more than one aircraft which, by prior arrangement between the pilots, operates as a single large, virtual aircraft with regard to navigation and position reporting. It has aroused great interest worldwide because of its great potential in military and civil applications. There are reasons to believe that formation-flying cooperative behavior can increase the efficiency of group performance. For example, UAVs flying in certain close formation would enjoy a benefit of substantial reduction in induced drag, resulting in 15% energy savings for downstream UAVs. In addition, military aircraft, ground units, and naval forces use this strategy to

Received 18 January 2017; Revised 18 December 2017; Accepted 18 December 2017; Published 6 February 2018. This paper was recommended for publication in its revised form by editorial board member, Biao Wang.
Email Address: ¶*tslangzy@nus.edu.sg*

benefit from mutual protection, concentration of offensive power, increase of robustness, reconfiguration ability, and simplification of control. The theoretic research in the field of cooperative control of multiple vehicles have also made great progress since the last decade.

However, there are still many challenging problems in cooperative UAV control which interest researchers around the world [3]. For example, formation control is an important issue in coordinated control for a group of UAVs/robots. In many applications, a fleet of UAVs are required to follow a predefined trajectory while maintaining a desired spatial pattern and velocity. As such, in order to verify the theoretical design of the formation control system, many research institutes and companies have developed experimental platforms for the demonstrations of cooperative control of multi-UAVs around the globe. Time-varying formation control problems for UAV swarm systems using switching interaction topologies were investigated [4]. They proposed a distributed time-varying formation protocol and demonstrated the flight using four quadrotors flying in formation. In particular, Temasek Laboratories at the National University of Singapore (TL@NUS) has also demonstrated the high-precision multi-UAV teaming as shown in Fig. 1.

In addition, Intel has just launched its specially designed drone and its software platform, the Intel Shooting Star, for formation flight. It is capable of automating the choreography process by using only images. Based on these images, the software is able to automatically calculate the minimum amount of drones needed and at the same time, outputs the optimal sequence of launching of drones to carry out the predefined path. In addition, the formation size can be easily expendable. Intel also made a Guinness World Record for having "The Most UAVs Airborne Simultaneously" with a fleet of 500 Intel Shooting Star drones as shown in Fig. 2.

The capability of multi-UAVs formation flight with a predefined path has been shown by the examples mentioned above. However, most of the formation flights show a lack of accuracy in each UAV's position as GPS solution is generally adopted. It is due to the fact that the average

Fig. 1. Circle and fountain pattern by TL's 16 UAV system.

Fig. 2. 500 pattern by Intel.

horizontal accuracy of a decent GPS receiver is about 2.168 m at 95% of the time. A solution to the inaccuracy problem would be to utilize an improved GPS solution, the so-called real-time kinematic (RTK) or differential GPS (DGPS) solution, which is also the focus of our development in TL@NUS.

In summary, our research covers the development of a robust and high accuracy formation flight, while ensuring a comprehensive safety standard operating procedure (SOP) is conducted with the flight. This paper documented the efforts made on the development of a 16 UAVs formation system using self-customized UAV, T-Lion, as shown in Fig. 3. All UAVs are able to robustly receive commands and send out status data to ground control station (GCS) wirelessly within a safety distance. In addition, a single UAV safety pilot has the highest priority to take-over the control of all 16 drones when needed.

Our work includes the following main contributions:

- Synchronized multi-UAV path planning and flight control. Accurate positioning system and algorithms to fuse the RTK GPS and inertial measurement unit (IMU) for navigation.
- An in-house developed GCS that could monitor and send commands to multiple UAVs with only one operator. The GCS system boasts additional redundancy by being hot-swappable with two sets of communication modules, and it is expandable to more UAVs.
- Hardware and software features such as secured and reliable communication capabilities, redundancy localization equipment and techniques, electromagnetic interference (EMI) shielding for sensitive equipment on the UAVs, built-in-tests (BIT) and in-flight checks.
- A comprehensive safety management plan with safety features that cover possible failures and implement a virtual geofence around the area of operation.

In the rest of the paper, the logic and algorithms that were developed and integrated for automating the multi-UAVs

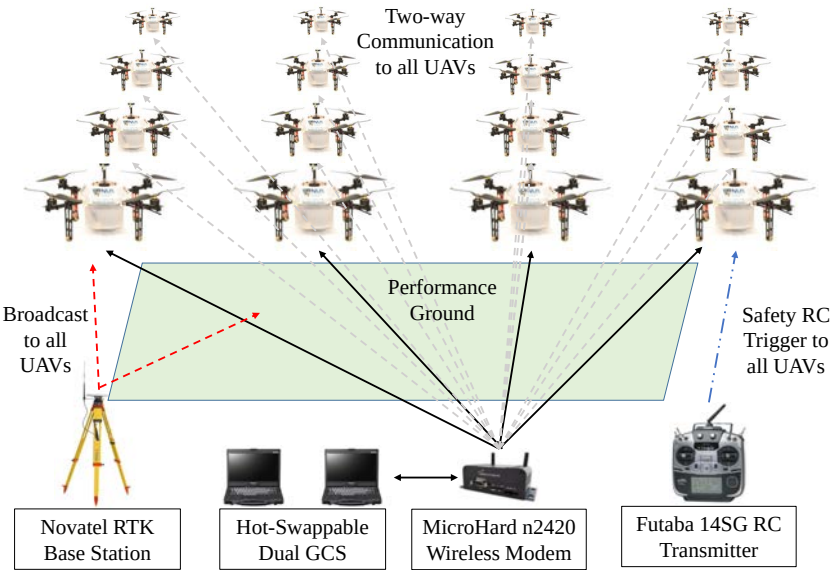

Fig. 3. Multi-UAV system configuration.

control and formation will be presented in the multiple sections below.

In Sec. 2, we will present the design of the navigation system based on RTK DGPS and IMU, accompanied by the experimental results for performance evaluation. Flight control of the UAV is covered in Sec. 3 based on the mathematical model of the hardware platform. Section 4 presents a minimum jerk trajectory approach for the multiple UAV formation, where both the minimum jerk trajectory and the flight time of the UAV will be optimized iteratively, until a converged local minimal solution is reached. Section 5 focuses on the construction of the hardware platform for the formation flight. The overall hardware development procedure is presented, including the bare platform, the avionics systems and onboard computer system integration. To achieve the goal of high reliability, stable communication, and low EMI in our system, we design our motherboard and low-level flight controller board. We will present environmental stress screening (ESS) tests for all our PCB boards to remove manufacturing workmanship defects and eliminate infant mortality failures prior to out usage.

A two-layer software system was also designed to perform robust and reliable flight missions. The implementation details of low-level and high-level autopilot will be described. A safety design approach with detailed failure condition considerations and precautions will be presented too.

In Sec. 6, an overall software system of the formation flight of UAV is proposed, which consists of two main parts: onboard system and GCS. The onboard system includes several sub-tasks, such as collect information from onboard sensors, compute the algorithms of coordination and control the UAV, and communicate with other onboard systems for coordination and with the GCS for monitoring purposes. The GCS is mainly used for monitoring the status of multiple UAVs, analyzing performance of control algorithm and coordination, etc. As a fully functional ground station, the GCS is able to communicate with multiple UAVs in real-time during flights. It also provides a graphic user interface (GUI) to receive commands from the operator and display the states of the UAVs in curve views, as well as indicate the UAVs in a geographical map.

In Sec. 7, a comprehensive safety system for UAV formation flight will be described that includes considerations from the hardware, software and electronics aspects of the UAV system. Algorithms are developed to prevent the UAV from escaping the safety zone and to let UAVs fly safely in every phase of the operation.

In Sec. 8, the real flight experiments are presented to verify the proposed formation control law in different manners. The formations performed included various patterns in continuous formation. From a circular Ferris-wheel pattern to star pattern and also a fountain pattern.

Finally, in Sec. 9, we draw the concluding remarks on topics that we worked on and plans for future development in the related work.

2. RTK DGPS Navigation System

In order to achieve centimeter-level positioning accuracy of the UAVs during the formation flight, we have utilized the

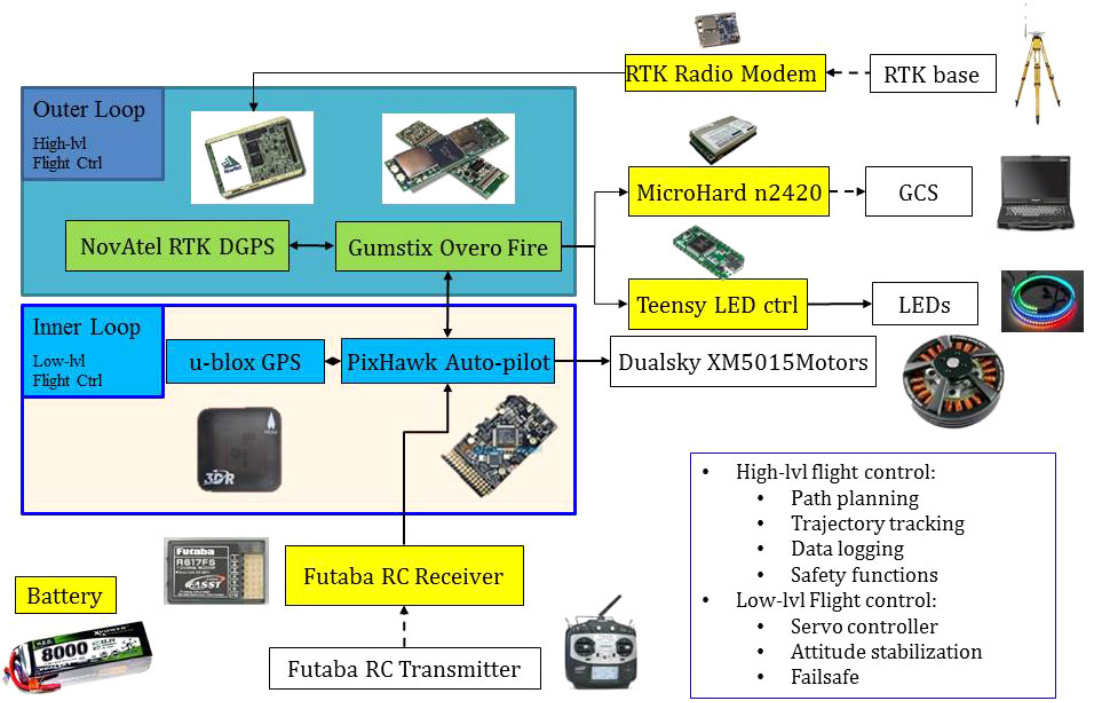

Fig. 4. Hardware configuration of the RTK system.

RTK DGPS as our main navigation system sensor, which is more accurate than DGPS. Although both of them use a fixed station to send correction signal, the main difference between them is that DGPS uses the pseudo-range correction, and RTK DGPS uses a carrier-phase correction.

RTK DGPS can provide very accurate position estimation in real-time, but its signal update rate is only about 10 Hz, which is much lower than the sampling rate of the position control loop of the UAV. In addition, the RTK position may have large errors if satellite signal is interrupted or blocked. Therefore, we have integrated RTK DGPS with an onboard IMU to obtain smooth position and velocity estimates with high signal rates.

The objectives of the proposed navigation system are to estimate the position and velocity of the UAV, and concurrently estimate the IMU measurement biases. We assume that the IMU measurements are corrupted by zero-mean Gaussian white noises and constant biases. Since the biases may vary every time the IMU is initialized, they must be estimated online and then compensated in the navigation algorithm. In fact, RTK-DGPS not only provides position information, but also velocity using the Doppler Effect. It is also assumed that those signals are corrupted by a zero-mean Gaussian white noise.

The RTK DGPS system includes a fixed base station, a rover and a radio modem. The whole system integrated with the UAV system is illustrated in Fig. 4. The structure of the navigation system is given in Fig. 5.

The measurements of the sensors are fused by a 9th-order Kalman filter (KF). The nine states of the KF are: 3-dimensional (3D) position, 3D velocity, and 3D acceleration bias. It is worth noting that the IMU measurement enters the KF through the process model, whereas the measurements of the RTK position and velocity enter the KF through the measurement model. There exist two update rates in the KF. The update rate of the process model (i.e. the IMU measurement) is 50 Hz, whereas that of the measurement model (i.e. the measurement of RTK) is 10 Hz. Denote T_s and T_v as the sampling periods of the processing and the measurement models, respectively. Then $T_s = 0.02$ sec and $T_v = 0.1$ sec.

2.1. *Sensor fusion: Process model*

Let $\boldsymbol{p}_n = [p_{n,x}, p_{n,y}, p_{n,z}]^T \in \mathbb{R}^3$ and $\boldsymbol{v}_n = [v_{n,x}, v_{n,y}, v_{n,z}]^T \in \mathbb{R}^3$, respectively, be the position and the velocity of the UAV in

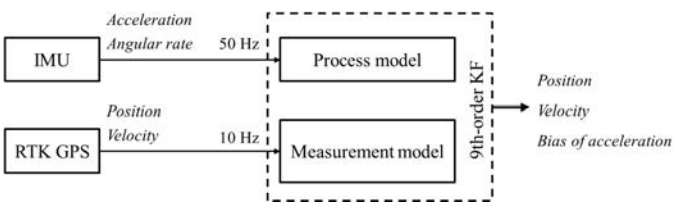

Fig. 5. RTK framework.

the navigation frame. The attitude represented by Euler angles (roll, pitch, and yaw) is denoted as $\rho = [\phi, \theta, \psi]^T \in \mathbb{R}^3$. Denote $\mathbf{e}_i$ with $i \in \{1,2,3\}$ as the ith column vector of the 3-by-3 identity matrix $\boldsymbol{I}_{3\times 3}$. The kinematic model of the UAV is

$$\begin{bmatrix} \dot{\boldsymbol{p}}_n \\ \dot{\boldsymbol{v}}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_n \\ \boldsymbol{R}_{n/b} a_{\text{nb}} + g\mathbf{e}_3 \end{bmatrix}, \tag{1}$$

where $\boldsymbol{a}_{nb} \in \mathbb{R}^3$ denotes the acceleration and angular rate of the UAV in the body frame; and g represents the local gravitational acceleration. The transformation matrix $\boldsymbol{R}_{n/b}$ is given by

$$\boldsymbol{R}_{n/b} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix},$$

where $s_* = \sin(*)$, $c_* = \cos(*)$, and $t_* = \tan(*)$.

Let $\boldsymbol{a}_{\text{nb,IMU}}$ be the acceleration measured by the IMU. Then we have,

$$(\boldsymbol{a})_{\text{nb,IMU}} = \boldsymbol{a}_{nb} - \boldsymbol{b}_a - \boldsymbol{w}_a, \tag{2}$$

where $\boldsymbol{w}_a \in \mathbb{R}^3$ are zero-mean Gaussian white noises; and $\boldsymbol{b}_a = [b_{a,x}, b_{a,y}, b_{a,z}]^{\mathrm{T}} \in \mathbb{R}^3$ are unknown but constant measurement biases. Since $\boldsymbol{b}_a$ may change every time the IMU is initialized, they must be estimated and compensated during the online calculation. To that end, the state vector is augmented by adding the unknown biases. From Eq. (1) and (2), the process model of the navigation system is obtained as

$$\begin{bmatrix} \dot{\boldsymbol{p}}_n \\ \dot{\boldsymbol{v}}_n \\ \dot{\boldsymbol{b}}_a \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_n \\ \boldsymbol{R}_{n/b}(\boldsymbol{a}_{\text{nb,IMU}} + \boldsymbol{b}_a + \boldsymbol{w}_a) + g\mathbf{e}_3 \\ \mathbf{0}_{3\times 1} \end{bmatrix}. \tag{3}$$

The process model Eq. (3) can be rewritten in a compact form as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u} + \boldsymbol{b} + \boldsymbol{w}), \tag{4}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_n \\ \boldsymbol{v}_n \\ \boldsymbol{b}_a \end{bmatrix}, \quad \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u} + \boldsymbol{b} + \boldsymbol{w}) = \begin{bmatrix} \boldsymbol{f}_p \\ \boldsymbol{f}_v \\ \boldsymbol{f}_{b_a} \end{bmatrix},$$
$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{a}_{\text{nb,IMU}} \end{bmatrix}, \quad \boldsymbol{b} = [\boldsymbol{b}_a], \quad \boldsymbol{w} = [\boldsymbol{w}_a].$$

2.2. *Sensor fusion: Measurement model*

The measurement model is based on the RTK position and the velocity is measured using relativistic Doppler Effect, which is given below.

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{p}_{n,\text{RTK}} \\ \boldsymbol{v}_{n,\text{RTK}} \end{bmatrix} = \boldsymbol{C}\boldsymbol{x} + n_{\text{RTK}}, \tag{5}$$

where

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{I}_{3\times 3} & \mathbf{0}_{3\times 3} \end{bmatrix}. \tag{6}$$

2.3. *Kalman filtering*

The details of the implementation of the Kalman filter are given below.

$$\begin{cases} \boldsymbol{x}(k+1) = \mathbf{A}(k)\boldsymbol{x}(k) + \mathbf{B}(\boldsymbol{u}(k) + \boldsymbol{w}(k)), \\ \boldsymbol{y}(k) = \mathbf{C}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k), \end{cases} \tag{7}$$

where $\boldsymbol{x} \in \Re^n, \boldsymbol{u} \in \Re^p$ and $\boldsymbol{y} \in \Re^m$ are state, input and measured variables. $\mathbf{A}(k), \mathbf{B}(k)$ and $\mathbf{C}(k)$ are system matrices with appropriate dimensions. $\boldsymbol{w} \in \Re^p$ and $\boldsymbol{v} \in \Re^m$ are input and measurement noise, which are zero-mean Gaussian noise. The objective of KF is to present $\hat{\boldsymbol{x}}(k|k)$ at the step k with the measurement $\boldsymbol{y}(k)$, input $\boldsymbol{u}(k-1)$, and the estimated $\hat{\boldsymbol{x}}(k|k-1)$. We need to assume that Eq. (8) is observable.

We have established the continuous process model in Eq. (4) and the measurement model in Eq. (5). We convert the model into the standard discrete-time version so that KF can be easily applied.

$$\begin{cases} \boldsymbol{x}(k+1) = \mathbf{A}(k)\boldsymbol{x}(k) + \mathbf{B}(\boldsymbol{u}(k) + \boldsymbol{w}(k)), \\ \boldsymbol{y}(k) = \mathbf{C}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k), \end{cases} \tag{8}$$

where

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{p}_n \\ \boldsymbol{v}_n \\ \boldsymbol{b}_a \end{bmatrix}, \quad \boldsymbol{y} := \begin{bmatrix} \boldsymbol{p}_{n,\text{RTK}} \\ \boldsymbol{v}_{n,\text{RTK}} \end{bmatrix}, \quad \boldsymbol{u} := \boldsymbol{R}_{b/n}^T \boldsymbol{a}_{\text{nb}} + g\boldsymbol{e}_3,$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & T_s\boldsymbol{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{I}_{3\times 3} & T_s\boldsymbol{I}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \boldsymbol{I}_{3\times 3} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times 3} \\ T_s\boldsymbol{I}_{3\times 3} \\ \mathbf{0}_{3\times 3} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{I}_{3\times 3} & \mathbf{0}_{3\times 3} \end{bmatrix}.$$

T_s is the sampling period. With the above model, the KF is applied to calculate $\hat{\boldsymbol{x}}(k+1|k)$ and $\hat{\boldsymbol{x}}(k|k)$. The design parameters of the KF are given by

$$\begin{aligned} \mathbf{Q} &= \text{diag}\{0.1,\ 0.1,\ 0.1,\ 1,\ 1,\ 1, 0.01,\ 0.01,\ 0.01\}, \\ \mathbf{R} &= \text{diag}\{0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01\}. \end{aligned} \tag{9}$$

$$\mathbf{P}(0) = \mathbf{BQB}^T, \quad \hat{\boldsymbol{x}}(-1) = \begin{bmatrix} \boldsymbol{p}_{n,\mathrm{RTK}}(0) \\ \boldsymbol{v}_{n,\mathrm{RTK}}(0) \\ \mathbf{0}^{3\times 3} \end{bmatrix}, \quad \boldsymbol{a}_{\mathrm{nb}}(-1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{10}$$

2.4. *Experimental results*

Here, we present flight experimental results to verify the effectiveness and robustness of the proposed navigation system. The filtered position results are shown in Fig. 6. The filtered velocity results are shown in Fig. 7. We can find the filtered signals are smoother than the raw data in presence of strong noise that results in abnormal measurements. The bias estimation is shown in Fig. 8, which varied during the experiment.

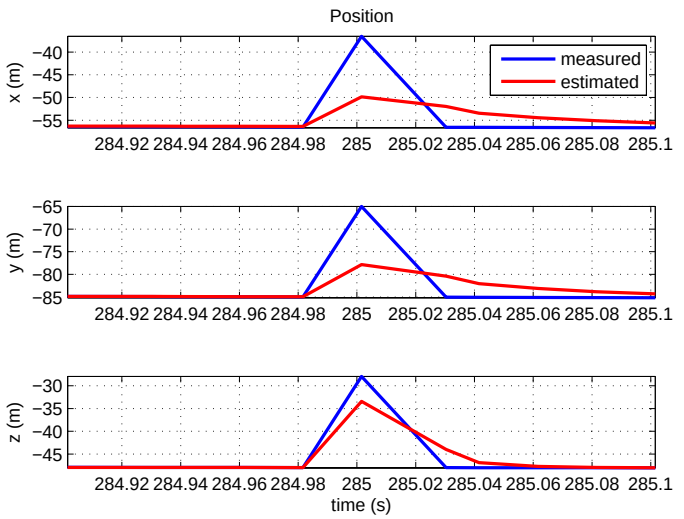

Fig. 6. Position estimation zoomed.

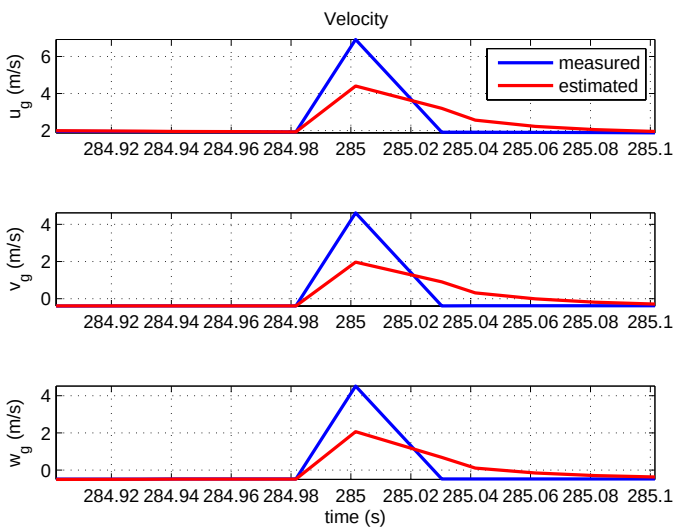

Fig. 7. Velocity estimation zoomed.

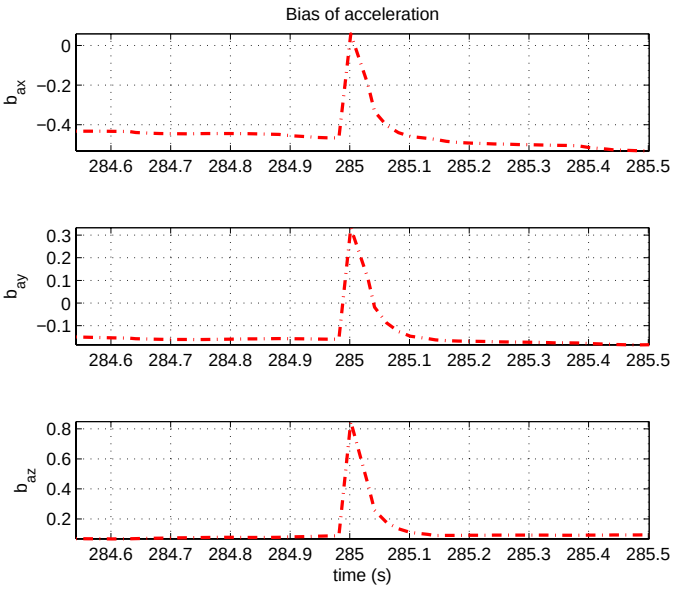

Fig. 8. Bias estimation.

3. Cascaded Control Structure

A generalized control structure was designed for use in our formation flight performance. Translation movement is the foundation for a useful vehicle and our design aims to track translational references as close as possible. First, a generalized point mass model for rotorcraft is given in Fig. 9. The model is expressed in a navigation frame $\mathcal{G}$ with the three axes defined as $x_{\mathbf{G}}$, $y_{\mathbf{G}}$, and $z_{\mathbf{G}}$, respectively. Since

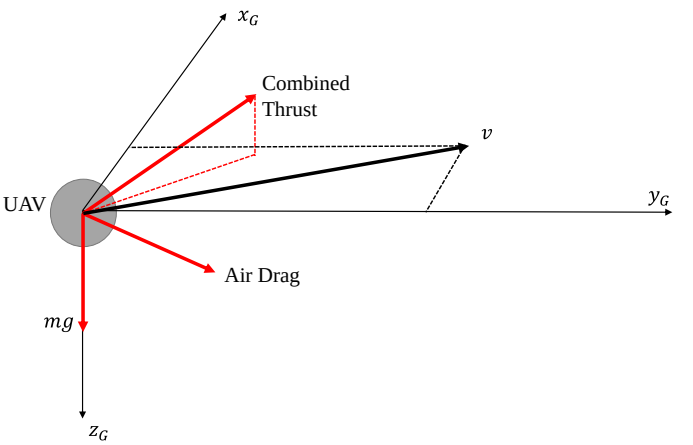

Fig. 9. Generalized point mass model for rotorcrafts.

rotorcrafts do not have wings, the main source of lift comes from their thrust. The three major forces acting on a rotorcraft are: combined thrust from all rotors, gravity and the drag force.

Such a model is expressed in Eqs. (11) and (12).

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{a} \\ \mathbf{a} = \dfrac{1}{m_q}(\mathbf{T}_\Sigma(\mathbf{q}) + \mathbf{d}_{\text{air}}(\mathbf{v}_{\text{air}})) + \mathbf{g}, \end{cases} \tag{11}$$

$$\begin{cases} \dot{\mathbf{x}}_q = \mathbf{f}_q(\mathbf{x}_q, \mathbf{v}) + \mathbf{g}_q(\mathbf{x}_q)\mathbf{u}_q \\ \mathbf{q} = C_q \mathbf{x}_q \end{cases}, \tag{12}$$

where $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}$ are the position, velocity and acceleration vector of the vehicle, m_q is the vehicle's mass, $\boldsymbol{T}_\Sigma, \boldsymbol{d}_{\text{air}}, \boldsymbol{g}$ are the three main forces of combined thrust, air drag and gravity. $\boldsymbol{v}_{\text{air}}$ is the velocity of air. Then, $\boldsymbol{x}_q$ is a collection of nontranslational states such as the attitudes of the vehicle, $\boldsymbol{u}_q$ is the true system inputs associated with physical actuators, $\boldsymbol{q}$ is a selected subset of $\boldsymbol{x}_q$ called controlled inner loop outputs and $\boldsymbol{f}_q, \boldsymbol{g}_q$ are functions describing the nonlinear dynamics. Here, the dynamics governed by Eq. (11) are called the translational model. The dynamics in Eq. (12) are called the attitude model as $\boldsymbol{x}_q$ usually contains parameters describing the rotorcraft's attitude. Quad-rotors are under-actuated systems, making them difficult to control in general. However, it is noticed that the attitude dynamics of rotorcraft are much faster than its translational dynamics. Therefore, the design of a cascaded controller utilizing the separation in timescale naturally emerged (see Fig. 10). The idea is to have an outer loop governing the position of the vehicle by manipulating $\boldsymbol{q}$. The required $\boldsymbol{q}$ (denoted as $\boldsymbol{q}_{\text{ref}}$) is then tracked by the attitude inner loop controller via $\boldsymbol{u}_q$. In other words, $\boldsymbol{q}$ is the controlled output for the inner loop system. Nevertheless, $\boldsymbol{T}_\Sigma(\boldsymbol{q})$ is commonly a nonlinear function. To design a proper outer loop controller, the procedure of nonlinear dynamic inversion (NDI) is adopted. For translational loop, the controlling target is naturally the position $\boldsymbol{p}$. Then, one has to repeatedly differentiate the output $\boldsymbol{p}$ until $\boldsymbol{q}$ appears. This happens when

$$\ddot{\boldsymbol{p}} = \frac{1}{m_q}(\boldsymbol{T}_\Sigma(\boldsymbol{q}) + \boldsymbol{d}_{\text{air}}(\boldsymbol{v}_{\text{air}})) + \boldsymbol{g}. \tag{13}$$

Considering the fact that $\boldsymbol{v}_{\text{air}}$ is difficult to measure, it is treated as disturbance which is neglected during dynamic inversion and handled by the robust controller. Thus, a virtual control input $\boldsymbol{u}_v$ is created as

$$\boldsymbol{u}_v = \frac{1}{m_q}(\boldsymbol{T}_\Sigma(\boldsymbol{q})) + \boldsymbol{g}, \tag{14}$$

and $\boldsymbol{q}_{\text{ref}}$ can be correspondingly found as

$$\boldsymbol{q}_{\text{ref}} = \boldsymbol{T}_\Sigma^{-1}(m_q(\boldsymbol{u}_v - \boldsymbol{g})). \tag{15}$$

Then, the control problem is reduced to that of designing a linear controller for a double integrator with virtual input

$$\ddot{\boldsymbol{p}} = \boldsymbol{u}_v, \tag{16}$$

and the real desired outer loop control input can be found in Eq. (15). Though not all controllers are structured in this way, the proposed structure has been successfully implemented on various types of rotorcrafts, such as helicopters [5], quad-rotors [6], octo-rotors [7], coaxials [8] and even unconventional vector thrust tail sitter [9] by our group. They share a similar translational controller since it is designed based on a double integrator model. However, their inner loop controller varies largely due to the differences in the inner loop dynamics described in Eq. (12).

The cascaded control structure of a quad-rotor is shown in Fig. 10. Note the heading angle reference of ψ_{ref} is also passed into the $\boldsymbol{T}_\Sigma^{-1}$ function. This is because the $\boldsymbol{q}$ of a

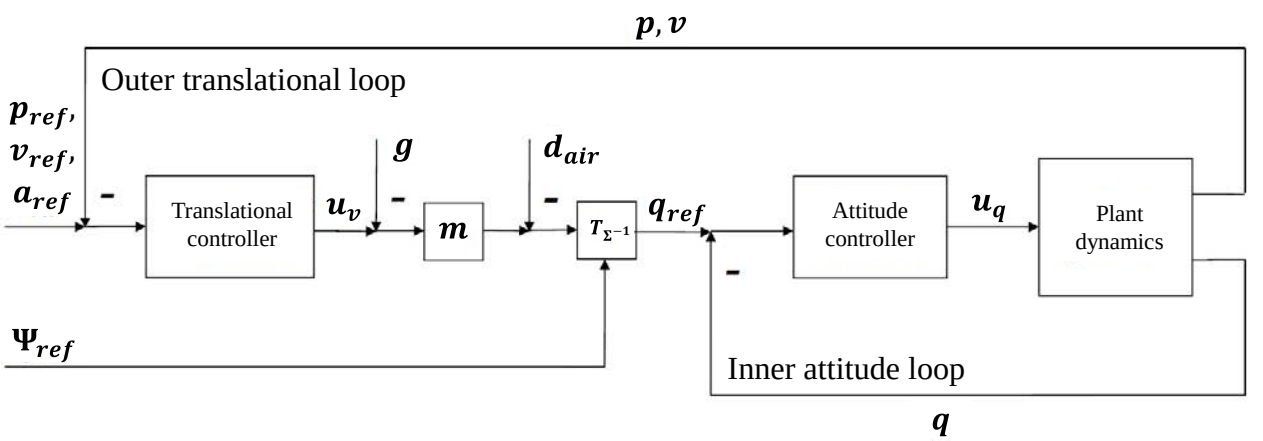

Fig. 10. Cascaded control structure for a quad-rotor.

quad-rotor's inner loop has four values $F_\Sigma, \phi, \theta, \psi$ which are the total thrust magnitude, roll, pitch and yaw angle, respectively. On the other hand, the virtual input for a 3D double integrator $\boldsymbol{u}_v = [u_{vx}, u_{vy}, u_{vz}]^T$ has only three values. In order to get a unique solution for $\boldsymbol{q}$ through $\boldsymbol{T}_\Sigma^{-1}$, the value of ψ is preset with ψ_{ref}.

3.1. *Robust perfect tracking (RPT) control*

After the dynamic inversion, the effective translational loop dynamics becomes a double integrator. From Eq. (16), it is clear that the 3 degrees of freedom (DOF) of the point mass model are decoupled. Then, it is natural to design a controller for each DOF separately. Here, the RPT controller from [10] is adopted for accuracy and robustness. Without input and states constraints, this controller can track any given reference with arbitrarily fast settling time in theory. For a linear time-invariant (LTI) system

$$\Sigma = \begin{cases} \dot{\boldsymbol{x}}_L = A\boldsymbol{x}_L + B\boldsymbol{u}_L + E\boldsymbol{w}_L \\ \boldsymbol{y}_L = C_1\boldsymbol{x}_L + D_1\boldsymbol{w}_L \\ \boldsymbol{h}_L = C_2\boldsymbol{x}_L + D_2\boldsymbol{u}_L + D_{22}\boldsymbol{w}_L, \end{cases} \tag{17}$$

with $\boldsymbol{x}_L, \boldsymbol{u}_L, \boldsymbol{w}_L, \boldsymbol{y}_L, \boldsymbol{h}_L$ being the state, control input, disturbance, measurement and controlled output, respectively. The RPT controller provides a dynamic measurement control law as follows

$$\begin{aligned} \dot{\boldsymbol{v}}_L &= A_c(\varepsilon)\boldsymbol{v}_L + B_c(\varepsilon)\boldsymbol{y}_L + G_0(\varepsilon)\boldsymbol{r}_L + \cdots + G_{\kappa-1}(\varepsilon)\boldsymbol{r}_L^{\kappa-1}, \\ \boldsymbol{u}_L &= C_c(\varepsilon)\boldsymbol{v}_L + D_c(\varepsilon)\boldsymbol{y}_L + H_0(\varepsilon)\boldsymbol{r}_L + \cdots + H_{\kappa-1}(\varepsilon)\boldsymbol{r}_L^{\kappa-1}. \end{aligned}$$

When a proper $\varepsilon > 0$ is chosen, the controller is then capable of

(1) Stabilizing the closed-loop system asymptotically, subjected to zero reference.
(2) If $e_L(t, \varepsilon)$ is the tracking error, then for any initial condition $\boldsymbol{x}_{L0}$, there exists

$$\|e_L\|_p = \left(\int_0^\infty |e_L(t)^p| dt\right)^{1/p} \to 0, \quad \text{as } \varepsilon \to 0. \tag{18}$$

The single-axis double integrator model can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i, \quad \boldsymbol{y} = \boldsymbol{x}, \tag{19}$$

where $\boldsymbol{x} = [p\ v]$ (p, v are the equivalents of $\boldsymbol{p}, \boldsymbol{v}$ on a specific axis, respectively), u is the virtual input (aka. single-axis acceleration) and $\boldsymbol{y}$ stands for the single-axis measurements. For better tracking performance, an error integral is introduced with an augmented system formulated as

$$\begin{cases} \dot{\boldsymbol{x}}_{\text{aug}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\boldsymbol{x}_{\text{aug}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_{\text{aug}} \\ \boldsymbol{y}_{\text{aug}} = \boldsymbol{x}_{\text{aug}} \\ h_{\text{aug}} = [-1\ 0\ 0\ 1\ 0]\boldsymbol{x}_{\text{aug}}, \end{cases} \tag{20}$$

where $\boldsymbol{x}_{\text{aug}} = [p_{\text{ref}}\ v_{\text{ref}}\ a_{\text{ref}}\ p\ v]^{\text{T}}$ with $p_{\text{ref}}, v_{\text{ref}}, a_{\text{ref}}$ as the position, velocity and acceleration references in the same axis of p, v. According to Ref. 11, a linear feedback law of the following form can be formulated as:

$$u_{\text{aug}} = F_{\text{aug}}\boldsymbol{x}_{\text{aug}}, \tag{21}$$

where

$$F_{\text{aug}} = \left[\frac{\omega_n^2}{\varepsilon^2}\ \frac{2\zeta\omega_n}{\varepsilon}\ 1 - \frac{\omega_n^2}{\varepsilon^2}\ -\frac{2\zeta\omega_n}{\varepsilon}\right].$$

The ε becomes a design parameter for adjusting the bandwidth of the closed-loop system. ω_n, ζ are the parameters that determine the desired pole locations of the infinite zero structure of (Eq. 20) through

$$p_{\text{character}}(s_L) = s_L^2 + 2\zeta\omega_n s_L + \omega_n^2. \tag{22}$$

s_L is the standard Laplace transformation symbol for complex numbers and this is used to determine pole location. Theoretically, it is possible to achieve arbitrarily fast response when ε is small enough. However, due to the requirements of time-scale separation of the inner and outer loop, it is wise to limit the bandwidth of the outer loop to be much smaller than that of the inner loop. The design procedure of the cascaded controller can be summarized as follows:

(1) Find inner loop controlled output $\boldsymbol{q}$ and its relationship to force $\boldsymbol{T}_\Sigma$.
(2) Design the inner loop controller and measure the closed inner loop's bandwidth.
(3) Perform dynamic inversion to simplify the nonlinear model, find virtual input and $\boldsymbol{T}_\Sigma^{-1}$.
(4) Design the outer loop controller based on the simplified linear model with a much smaller bandwidth compared to the inner loop.

Based on the dynamic properties of our platform, a cascaded controller is implemented to track the reference generated by the guidance unit. The cascade control structure is successfully implemented on various vehicles in our group. Further, the translational controller is also generalized because the outer loop mode of the vehicle is simplified to a double integrator after dynamic inversion. The

proposed system has been used in various competitions and projects [6,12,13] and is currently used as well in our formation flight.

4. Trajectory Generation Algorithm

For acquiring references for the multi-drone performance, a trajectory designing toolbox was developed combining several different methods. It consists of the minimum jerk trajectory generator, the reference command filters and a jerk limited two-point boundary value problem (TPBVP) solver [14]. It allows the user to input a series of way-points and adjust the smoothness of the trajectory as well as the maximum flying velocity. It serves as a useful tool for developing and testing various guidance, control and localization algorithms as well as platforms. In this paper, the representative algorithms like the B-spline minimum jerk trajectory generation, the time optimal jerk limited TPBVP and the coordinate rotation methods will be discussed.

4.1. *Dynamic feasibility*

By assuming faster dynamics of the inner attitude loop, the dynamics of the system is guaranteed by satisfying the magnitude and rate limits of $\boldsymbol{T}_\Sigma$. By assuming no environmental wind ($\boldsymbol{v}_{\text{air}} = -\boldsymbol{v}$), it gives

$$\boldsymbol{T}_\Sigma(\boldsymbol{q}) = m_q(\ddot{\boldsymbol{p}} - \boldsymbol{g}) - \boldsymbol{d}_{\text{air}}(-\dot{\boldsymbol{p}}). \tag{23}$$

The magnitude and rate of $\boldsymbol{T}_\Sigma$ can be limited by introducing constraints on the derivatives of the position.

4.2. *Minimum jerk trajectory*

From Eq. (23), it shows the changing rate of $\boldsymbol{T}_\Sigma$ is related to the jerk of the trajectory. Therefore, for a smooth flight, a minimum jerk trajectory with constrained derivatives can be considered. To construct such a reference, a cubic clamped normalized uniform B-spline (NUBS) is used to synthesize the user input which does not satisfy the vehicle dynamics. The proposed method is based on the optimal smoothing and interpolating method in Refs. 15 and 16 and an additional time vector optimization search.

The clamped cubic NUBS has the following form

$$S_3(s) = \sum_{i=-2}^{0} \boldsymbol{c}_i N_{i+2,3}(s) + \sum_{i=1}^{M} \boldsymbol{c}_i B_3(s-i+1) + \sum_{i=M+1}^{M+3} \boldsymbol{c}_i N_{i+2,3}(s), \tag{24}$$

where

$$B_3(\varpi) = \begin{cases} \frac{1}{6}\varpi^3, & \text{if } \varpi \in [0,1), \\ \frac{1}{6} - \frac{1}{2}(\varpi-1)^3 + \frac{1}{2}(\varpi-1)^2 + \frac{1}{2}(\varpi-1), & \text{if } \varpi \in [1,2), \\ \frac{1}{6} + \frac{1}{2}(\varpi-3)^3 + \frac{1}{2}\varpi - 3)^2 - \frac{1}{2}(\varpi-3), & \text{if } \varpi \in [2,3), \\ -\frac{1}{6}(\varpi-4)^3, & \text{if } \varpi \in [3,4), \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

and

$$N_{0,3}(\varpi) = \begin{cases} (1-\varpi)^3, & \text{if } \varpi \in [0,1), \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

$$N_{1,3}(\varpi) = \begin{cases} \varpi(1-\varpi)^2 + \frac{\varpi(4-3\varpi)(2-\varpi)}{8}, & \text{if } \varpi \in [0,1), \\ -\frac{(\varpi-2)^3}{4}, & \text{if } \varpi \in [1,2), \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

$$N_{2,3}(\varpi) = \begin{cases} \frac{\varpi^2(-3\varpi+4)}{4} + \frac{\varpi^2(3-\varpi)}{6}, & \text{if } \varpi \in [0,1), \\ \frac{\varpi(\varpi-2)^2}{4} + \frac{(3-\varpi)(-2\varpi^2+6\varpi-3)}{6}, & \text{if } \varpi \in [1,2), \\ \frac{\varpi^2(-3\varpi+4)}{4} + \frac{\varpi^2(3-\varpi)}{6}, & \text{if } \varpi \in [2,3), \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

while

$$\begin{aligned} N_{M+3,3}(\varpi) &= N_{2,3}(-\varpi + M + 3), \\ N_{M+4,3}(\varpi) &= N_{1,3}(-\varpi + M + 3), \\ N_{M+5,3}(\varpi) &= N_{0,3}(-\varpi + M + 3). \end{aligned} \tag{29}$$

The parameter s is the path parameter, its relationship to time t is

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \alpha. \tag{30}$$

Let

$$\Gamma_{i,3}(s) = \begin{cases} N_{i,3}(s), & \text{if } i \in \begin{Bmatrix} 0, 1, 2, M+3, \\ M+4, M+5 \end{Bmatrix}, \\ B_3(s - i + 3), & \text{otherwise.} \end{cases} \tag{31}$$

with which the clamped cubic NUBS can be written as

$$S_3(s) = \sum_{i=0}^{M+5} \boldsymbol{\tau}_i \Gamma_{i,3}(s), \quad \boldsymbol{\tau}_i = \boldsymbol{c}_{i-2}, \tag{32}$$

where $\boldsymbol{\tau}$ is a shifted representation of $\boldsymbol{c}$. Here, $\Gamma_{i,3}(s)$ is the base and $\boldsymbol{\tau}$ is the control point. For a given set of data

$$\begin{aligned} D_s &= [d_0, d_1, \ldots, d_{I_\mathrm{d}}]^T, \\ D_\mathrm{time} &= [t_0, t_1, \ldots, t_{I_\mathrm{d}}]^T. \end{aligned} \tag{33}$$

The minimum jerk interpolation problem is to minimize

$$J_s = w_\mathrm{g} \int_{-\infty}^{\infty} j_s^2(t) dt + \sum_{i=1}^{I_\mathrm{d}} (S_3(\alpha t_i) - d_i)^2, \tag{34}$$

where

$$j_s(t) = \sum_{i=0}^{M+5} \frac{d^3}{dt^3} \tau_i \Gamma_{i,3}(s), \quad \tau_i \in \mathbb{R}. \tag{35}$$

The optimization problem can be transferred into a quadratic programming following the technique in [15] as

$$J_{s\min} = \min_{T_s} \{T_s^T (w_g G_s + H^T H) T_s - 2 D_s^T H \tau + D_s^T D_s\}. \tag{36}$$

However, since the clamped spline is used, the formulation of G_s and H is slightly different. Moreover, the limits on derivatives can be introduced as linear nonequality constraints as discussed in Ref. 17. Sometimes, the user input only contains the geometric information of D_s but the time information D_time is lacking. A gradient descent with backtrack line search is performed to find the best time vector D_time to finish the geometrical path.

4.3. *Minimum time trajectory*

On the other hand, during the trajectory designing process, time optimal trajectory is preferred. In the designing toolbox, the jerk limited TPBVP solver from [14] is adopted. It utilizes the bang-zero-bang control idea for switching the jerk to produce states limited trajectory. The problem to be solved is shown as

$$\begin{aligned} &\min_{u_j(t), t \in [0,T]} \left\{ J = \int_{t=0}^{T} dt \right\} \; s.t. \\ &p(0) = p_0, \quad p(T) = p_\mathrm{ref} \\ &v(0) = v_0, \quad v(T) = v_\mathrm{ref} \\ &a(0) = a_0, \quad a(T) = 0 \\ &\dot{p}(t) = v(t) \\ &\dot{v}(t) = a(t) \\ &\dot{a}(t) = u_j(t) \\ &-v_\mathrm{max} \leqslant v(t) \leqslant v_\mathrm{max}, \qquad \forall t \in [0, T] \\ &-a_\mathrm{max} \leqslant a(t) \leqslant a_\mathrm{max}, \qquad \forall t \in [0, T] \\ &-u_{j\mathrm{max}} \leqslant u_j(t) \leqslant u_{j\mathrm{max}}, \qquad \forall t \in [0, T]. \end{aligned} \tag{37}$$

The problem can be solved by designing a feedback control law such as a command filter. However, for reference designing, it is desired to generate the trajectory in one run. As shown in Ref. 18, the problem could be solved using seven motion segments where each of them can be described by

$$\begin{aligned} Rp(t, p_0, v_0, a_0, u_j) &= p_0 + v_0 \cdot t + \frac{1}{2} a_0 \cdot t^2 + \frac{1}{6} u_j \cdot t^3 \\ R_v(t, v_0, a_0, u_j) &= v_0 + a_0 \cdot t + \frac{1}{2} u_j \cdot t^2 \\ R_a(t, a_0, u_j) &= a_0 + u_j \cdot t. \end{aligned} \tag{38}$$

These seven segments can be categorized into

(1) Velocity increasing (VI) phase.
(2) Velocity constant (VC) phase.
(3) Velocity decreasing (VD) phase.

Here, the VI and VD phases each take three segments and the VC phase has one segment. A typical 3-phase-7-segment trajectory is illustrated in Fig. 11.

The overall TPBVP can be written as

$$\begin{aligned} p_{i+1} &= R_p(\Delta t_{i+1}, p_i, v_i, a_i, u_i), \quad \forall i \in [0, 5] \\ v_{i+1} &= R_v(\Delta t_{i+1}, v_i, a_i, u_i), \quad \forall i \in [0, 5] \\ a_{i+1} &= R_a(\Delta t_{i+1}, a_i, u_i), \quad \forall i \in [0, 5] \\ p_\mathrm{ref} &= R_p(\Delta t_7, p_6, v_6, a_6, u_6) \\ v_\mathrm{ref} &= R_v(\Delta t_7, v_6, a_6, u_6) \\ 0 &= R_a(\Delta t_7, a_6, u_6). \end{aligned} \tag{39}$$

Due to the requirements of time optimality, the magnitude of $u_i, i \in [0, 6]$ is either u_max or 0. Therefore, once its sign is determined so is its value. For each phase, the

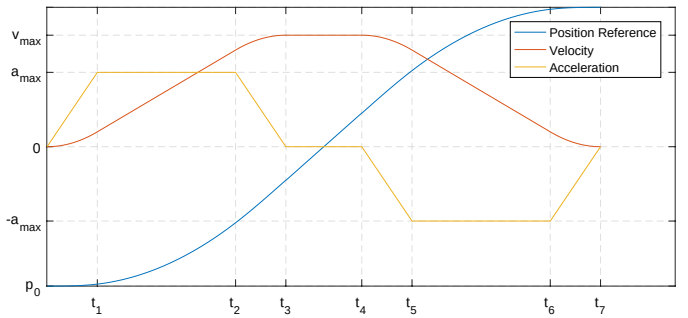

Fig. 11. Seven segments of the point-to-point maneuver.

acceleration profile could be either wedge or trapezoidal shaped. Based on the initial condition, the VI phase might also become a VD phase. As shown in Refs. 18 and 19, the TPBVP can be categorized based on the property and shape of the acceleration profile in each phase. For each case, the sign of $u_i, i \in [0, 6]$ can be determined, and the functions in Eq. (39) can be solved to give the final trajectory.

To qualitatively determine the shape and case of the acceleration profile, a decision tree-based pruning method described in Ref. 19 can be applied. The basic idea is to prune the area covered by the acceleration profile during VI and VD phases until the final stopping point no longer overshoots the target.

For multi-dimensional case, it is sometimes desirable to have all the axes reaching their final states all at the same time. This is achieved by extending the reaching time of each axis to a common value usually following the dimension with longest reaching time. However, the inaccessible time region covered in Ref. 18 needs to be considered during the process.

4.4. *Coordinate projection*

Through using the time or phase synchronization techniques in Refs. 18 and 20 a straight line path can be achieved in 3D space, for more complex patterns, a coordinate projection strategy can be adopted for the ease of designing. Consider the case of following a line segment path in Fig. 12.

A new coordinate $\mathcal{W}$ is built with its origin at WP_i and its x-axis pointing towards WP_{i+1}. With this coordinate, the path following problem can be solved by

(1) In the x-axis, steer the vehicle to $x_{\mathrm{ref}} = \|\mathrm{WP}_i - \mathrm{WP}_{i+1}\|$.
(2) In the y-axis, steer the vehicle to $y_{\mathrm{ref}} = 0$.

With multiple way points, a reaching distance could be set to construct more complex paths. Figure 13 shows a 2D path with the tracking simulation using the model of a real quadrotor.

Moreover, by using a polar coordinate, circle and spiral reference can also be generated. The limits on angular

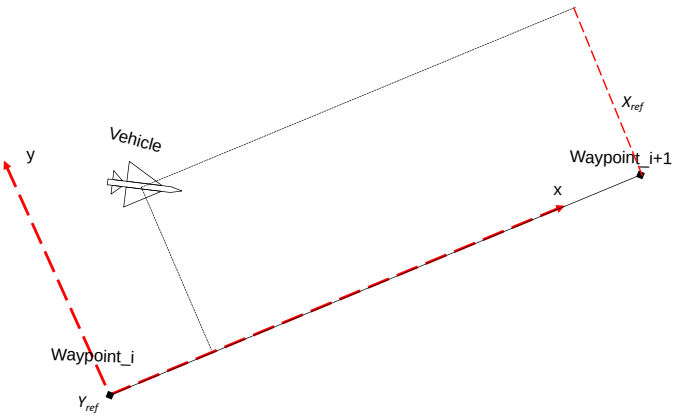

Fig. 12. Coordinates following 2D path.

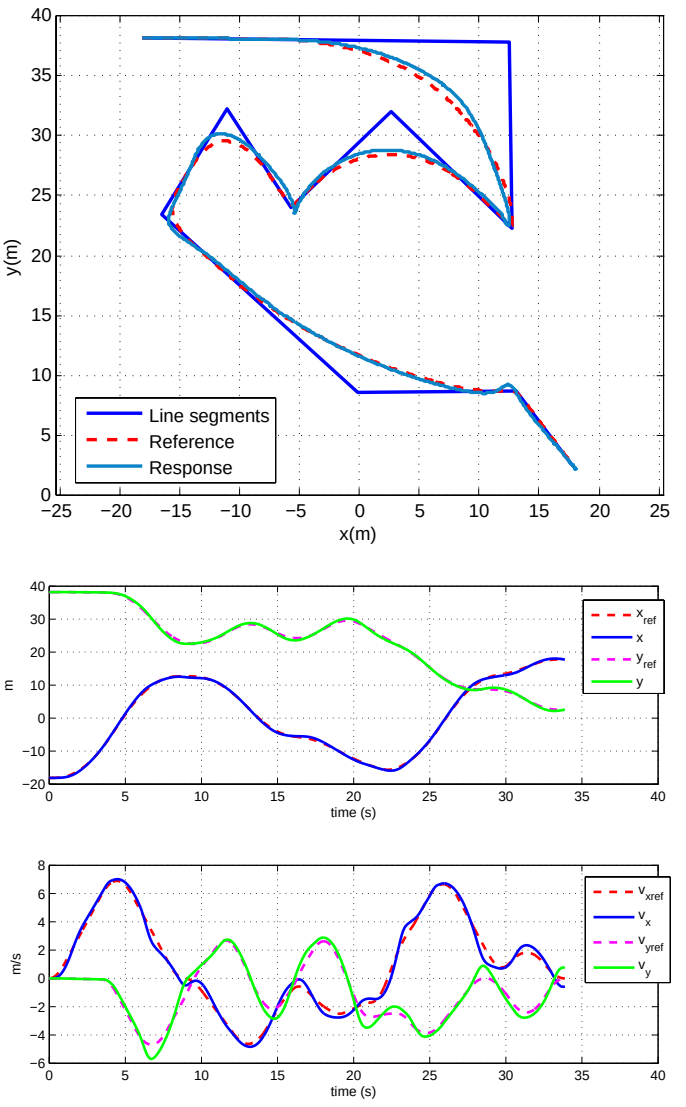

Fig. 13. Following of complex path using the position command filter.

speed, angular acceleration and angular jerk shall be calculated so that they do not violate the vehicle's dynamics.

4.5. *Collision avoidance*

The UAV system described in this paper does not have any communication established between individual UAV agents. As such, we could not implement any form of dynamic obstacle avoidance algorithms. Since the UAV formation system was designed to be centrally controlled and all agents have their paths pre-planned offline, through the evaluation of fault-tree analysis in our safety system described in Sec. 7, we deduced that we do not need an online obstacle avoidance algorithm present in the system. This was due to the fact that should the UAVs be close to collision, there could be failures in either one of the two parts shown below:

(1) Failure in GPS module/Bad GPS signals.
(2) Failure in trajectory tracking/Not able to track given path closely.

These failure points are covered in our safety contingencies where both failures will result in the UAVs performing "safety landing" on the spot either by GPS (if GPS is available) or performing "safety descending" by IMU integration in the short period of time needed to perform landing.

However, this does not mean that there are no forms of obstacle avoidance present in this paper. The obstacle avoidance exists during the designing of the offline trajectory.

During the design of the offline trajectory of all the UAVs, particular attention was paid to the path assignment problem to ensure that there would not be any collisions among UAVs. The two-layer strategy that we used was to implement the Munkres assignment algorithm [21] in associating which UAVs are assigned to which way-points based on their relative distances to the target way-points with a constraint on the inter-UAV distances that calls the reassignment algorithm again if violated. Through this assignment algorithm, the UAV with the shortest direct relative distance to the target position is always assigned to it. When running this assignment algorithm, the constraint on inter-UAV separation is set at 10 m.

5. UAV Platform for Formation Flight

In this section, we look into the development of a vertical take-off and landing (VTOL) UAV platform that will be eventually used to implement algorithms required by the UAV light show. We need to first identify the most demanding requirements based on the consideration of physical limitations of the platform which are namely types of UAV, size and weight. Finally, this boils down to three fundamental hardware requirements on the bare UAV platform, which are listed as follows:

- Size appropriate to fly in outdoor and urban environments.
- Additional payload (payload excluding bare platform and battery) no less than 1.5 kg
- Flight endurance of no less than 25 min with payload, such as LED lights.

The multi-rotor design was chosen with the aim for a robust and reliable platform that could perform mass formation flight in a dedicated 3D space. This marries the implementation of hardware design, avionics design and onboard software development to achieve a high-performance UAV that is capable of such a feat. The hardware design was researched over a span of six months to arrive at a simple yet efficient design that was hardy and yet not very heavy. The platform could achieve a high payload and long endurance flight using high-capacity 6-cell batteries. The avionics block was designed to be integrated and compact. It includes an autopilot system to manage the overall UAV performance and advance IMU sensors and outer-loop computer to manage the trajectory planning portion. The whole avionics block is then mounted on vibration isolators to reduce the vibration noise caused by the propellers when the UAV is in flight. Lastly, to allow users to have an overall control over all the UAVs in its formation, it is necessary to develop the GCS software. The GCS software allows the users to monitor all UAV health statuses as well as all sensor readings. The GCS is also capable of displaying the flight trajectory of all the UAVs over a Google map of the flight zone. Other than the monitoring capabilities, the GCS software allows the user to send commands such as take-off, landing and way-point flights through the onboard data-link provided by the MicroHard wireless data system.

5.1. *Choice of platform type*

With these requirements in mind, two platforms are taken into consideration, namely a quadrotor platform and a hexrotor platform. In general, a hexrotor has better payload capacity, while a quadrotor is more energy efficient provided that they have more or less the same weight and size. However, due to practical problems such as the availability of commercial-off-the-shelf (COTS) components (motors, electronic speed controllers (ESCs), propellers), the prototype quadrotor (see Fig. 14) and hexrotor (see Fig. 15) platforms are a bit different in weight and size.

Nevertheless, endurance tests have been carried out on both platforms with two kinds of payload options (1.5 kg and 2.0 kg). The testing show that both platforms can meet the 20 min minimum endurance requirement. The hexrotor

Fig. 14. A customized quadrotor platform.

Fig. 15. A customized hexrotor platform.

performs better in endurance, but at the price of larger dimension and heavier total weight. By considering the future hardware maintenance and flight test convenience, quadrotor platform is finally chosen.

After implementing our avionics system, redundancy system, flight control system as well as designing a semi-waterproof enclosure, we have our final UAV platform, T-Lion, as shown in Fig. 16.

5.2. *Avionic system*

To realize fully autonomous flight, an avionic system has been developed. All the components of the avionic system are the most suitable COTS products to date. The details and usage of each component in the avionic system are presented in the following parts.

Fig. 16. T-Lion UAV platform.

5.2.1. *Navigation sensors*

In order to achieve centimeter positioning accuracy of the UAVs during the formation flight, we have utilized the RTK GPS in the navigation system, which is more accurate than conventional differential GPS (DGPS). Although both of them use a fixed station to send correction signal, the main difference between them is that DGPS uses the pseudo-range correction, and RTK GPS uses a carrier-phase correction.

RTK GPS can provide very accurate position estimation in real time, but its signal update rate is about 10 Hz, which is much lower than the sampling rate of the position control loop of the UAV. In addition, the RTK position may still have erroneous readings if the satellite signal is interrupted. Therefore, we have integrated RTK GPS with onboard IMU to obtain high-signal rate and smooth position and velocity estimation.

The objectives of the proposed navigation system are to estimate the position and velocity of the UAV, and concurrently estimate the IMU measurement biases. We assume that the IMU measurements are corrupted by zero-mean Gaussian white noise and constant biases. Since the biases may vary every time the IMU is initialized, they must be estimated online and then compensated in the navigation algorithm. In fact, RTKGPS not only provides position information, but also velocity using Doppler Effect. It is assumed that those signals are corrupted by a zero-mean Gaussian white noise.

5.2.2. *Communications*

There are multiple wireless communications on the UAV, and each of the communication links must be correct and robust. Thus, we need to make sure the interference is as low as possible on the frequencies we used. In our UAV, the

following four frequencies are used:

- RC link: Center frequency: 2.437 GHz, hopping;
- Microhard data link: Center frequency: 5.769 GHz, hopping;
- RTK link: Center frequency: 457.225 MHz;
- GPS signal: 1.5 GHz.

When considering the communications to reduce the interference among the frequencies we used, the four frequencies selected are as far as possible among each other. And then, during the performance, before powering on our equipment, we will measure the ambient noise first to make sure it is clean on the frequencies we used. Then, after powering on all the communication links, we will keep monitoring the signals.

Another significant interference we noticed is that the GPS signal is interfered by the onboard computers on the UAV. At the beginning stage, we have to mount the GPS antenna as high as possible to reduce the interference which resulted in a significantly improved signal performance. Then, a copper plate was mounted between the GPS antenna and onboard computers. We found that the majority of the frequency noise produced by onboard computers has been filtered by this hardware improvement. A copper box was eventually used to shield the GPS receiver; and a high-frequency EMI suppression ferrite bead is used on the GPS antenna cable. After these improvements, the GPS receiving strength was significantly improved even with a lowered GPS antenna, the satellites received weren't reduced due to the noise produced by onboard computers.

5.2.3. *Control Hub*

Control Hub is a motherboard designed to host subsystems for control purposes. It has the following features:

- ***Module connection.*** Aforementioned modules, such as the Gumstix board, is installed on the slots on Control Hub and connected to the onboard power regulator and other essential components through the Control Hub. Besides the mounting slots, extra mounting holes on Control Hub have been used to lock the installed modules to resist the vibration and shock in flight and landing. Manual wire wrap has been minimized to improve reliability and quality of the system.
- ***Level shifter.*** An onboard level shifter: MAX3232 has been built in Control Hub to convert the serial signal from RS-232 level to TTL level, which has been used to make the output of RTK GPS compatible with the Gumstix board.
- ***Power regulation.*** To power up all the avionics, linear regulators are built in Control Hub to convert a power input from a 6-cell LiPo battery into a 5 V output with 10 A capacity and a 2–8 V adjustable output with 10 A capacity. The 5 V output powers the Gumstix board, the rate gyro and the electronic mixer. The adjustable output powers the servos.

6. Onboard Software System Design

The software system running on the UAV onboard system is critical for the normal operations of the UAVs during the flight. The software system should be able to handle not only automatic flight control but also contingency cases during the flight as well to avoid unnecessary accidents. The onboard software is responsible for sensor data collection, sensor data processing, mission logic decision and flight control. All the functionality needs to be executed reliably in a real-time fashion and thoroughly tested in both simulation and practical situations.

The two-layer software structure is illustrated in Fig. 17. The two layers of autopilot software systems share the same modular design with modules such as sensors, estimation, mission planning, outer-loop control and inner-loop control. The two autopilot system runs in parallel simultaneously with the *Mission planning* module in the *Fundamental autopilot* selecting the control signals. The *Fundamental autopilot* is controlled by the ground safety pilot handling the fail-safe transmitter.

The flight control of the UAVs consists of three modes: manual mode, semi-autonomous mode and autonomous mode. When in manual mode, the transmitter will generate the control signals to the *Inner-loop control* and the pilot has full control of the UAVs during the flight. In autonomous mode, the control signals generated from the high-level autopilot will override the low-level autopilot during the whole flight. The autonomous behaviors such as take-off, performing path and landing will be scheduled in the *Mission planning* on the *High-level autopilot*. Another mode called semi-autonomous flight is also developed onboard. In semi-autonomous mode, the low-level autopilot will take over the flight control enabling the ground pilot to control multiple UAVs in stable attitude mode concurrently. The three-mode switch is triggered by one toggle button on the transmitter.

In both autopilots, the software systems are equipped with basically the same modules. With *Sensing*, the IMU data and GPS data are retrieved and processed in the *Estimation* module. In *Mission planning*, the low-level autopilot will handle the transmitter input to select the corresponding flight mode. Mission logic such as take-off, fly to rendezvous point, perform path, return home, safety trigger and landing are all coordinated within this module. *Outer-loop control* will generate the references for *Inner-loop control* to control the motors based on the flight mission.

The two autopilot system communicate with each other on a serial port based on mavlink protocol.

The software development of the two-level autopilot system is implemented on embedded processors with RTOS. The low-level autopilot software system is developed based on the wildly adopted open source project called Pixhawk. The Pixhawk flight stack is light weight yet with rich features such as convenient inter-process communication, online parameter update and so on. It is hosted on the 32-bit STM32F427 Cortex M4 processor and another 32-bit STM32F103 failsafe co-processor with Nuttx operating system. Our in-house developed control laws, semi-autonomous mode capability and inter-processor serial communication are developed based on the stable branch from Pixhawk. The other high-level autopilot is implemented on powerful OMAP3530 based computer module called Gumstix. The QNX 6.5.0 RTOS board support package (BSP) is customized with rich peripheral support and installed on the Gumstix. The high-level autopilot adopts a multi-thread architecture to coordinate the tasks in Fig. 17 to be executed in user-defined order.

6.1. *Formation flight implementation approach*

To facilitate the development and deployment of multiple UAV platforms, the onboard flight control software system is designed to be identical. Meanwhile, each onboard SD card stores the UAV ID, flight control parameters and predefined flight paths which can be loaded when the onboard program starts. The onboard configuration file will be modified over the development time while the onboard software system remains the same.

6.1.1. *Coordinate synchronization*

As the team formation flight is performed based on a unified local North-East-Down (NED) frame, the initial local NED frame needs to be synchronized before tracking the formation path. About 16 UAVs will be placed on the test field with one UAV as the leader. Once all the onboard systems have started the program, a synchronization command is applied to broadcast the leader's UAV GPS coordinate and heading readings to all the followers such that all

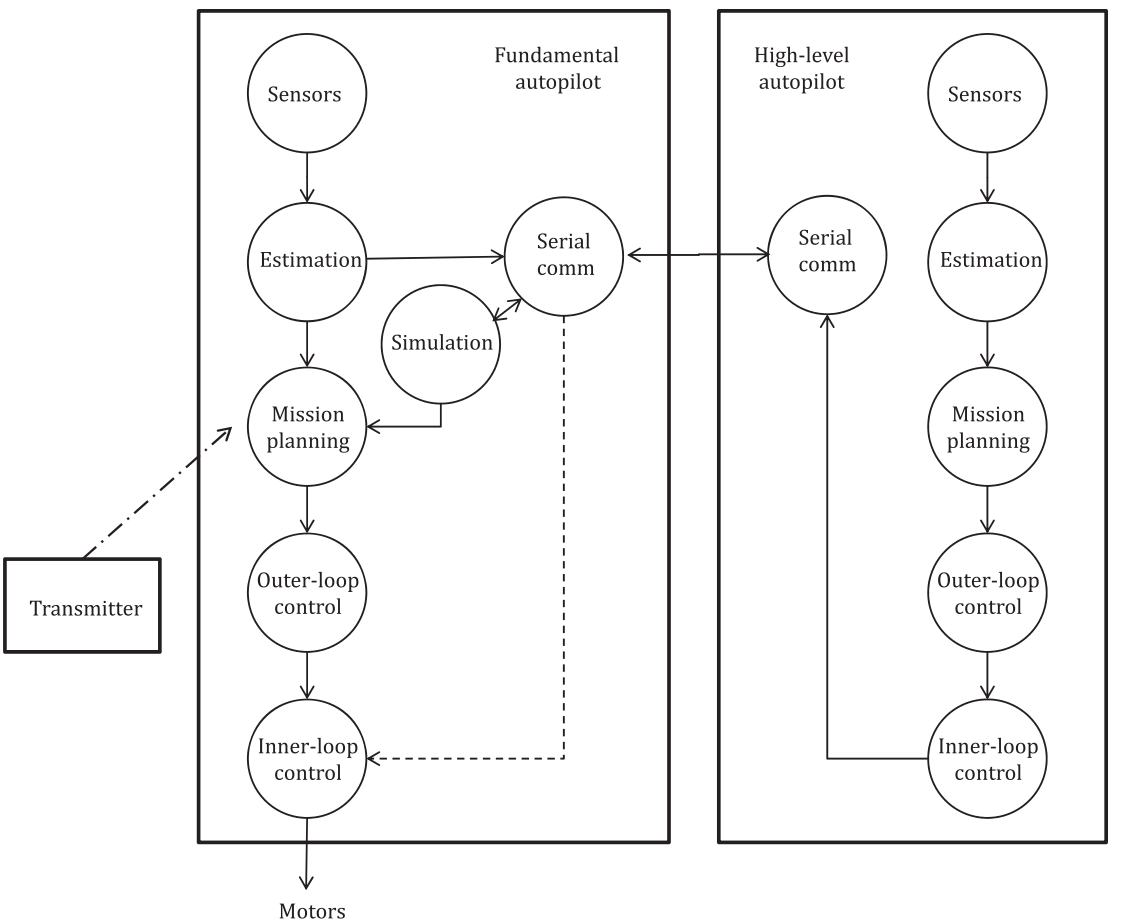

Fig. 17. Software structure of UAV onboard system.

the UAVs will re-establish their own local NED frame based on the leader's. All other UAVs will be placed in the desired offset position with respect to the leader. A feature is implemented onboard to automatically adjust followers' position during the take-off phase. When the take-off task is done, all the followers should be hovering at the position with the desired offset and same heading angle as the leader. Subsequently, the formation path command will be issued to each UAV to start the team formation.

6.1.2. *Time synchronization*

All the pre-defined formation paths will be started exactly at the same time to ensure the overall pattern consistency and can be easily visualized. The GPS Coordinated Universal Time (UTC) is obtained via the onboard GPS unit and used to trigger the execution of onboard paths. Currently, the time synchronization has been achieved in second level: e.g. the current time is year 2015, month 02, hour 03, minutes 35, seconds 40, nanoseconds 150000, the specific time to start the formation can be set at minute 36, so after hovering another 20 seconds, all the UAVs will start tracking the desired path and the overall team formation will be realized given the excellent onboard flight tracking control performance.

6.2. *MATLAB simulation*

We implement our off-line path planning algorithm into a MATLAB program and perform our 16 UAV synchronized path planning offline using it. Our program is able to ensure smooth continuous path planning at 50 Hz with constraints in velocity and accelerations implemented. The MATLAB program helps in the choreographing of the UAV formation as well as manages the transition of the UAVs. It also shows the user a 3D display of the whole formation flight before finally generating 16 path files with individual UAV flight paths.

In Fig. 18, you can observe the MATLAB simulation display (top row) compared with the real-life actual flight (bottom row) performed by our 16 UAVs. This MATLAB simulator provides us with a preview and comprehensive check on the performance of our 16-UAV formation prior to flying the UAVs in real-life.

6.3. *Hardware-in-the-loop simulation*

A simplified *Simulation* module is also integrated into the low-level autopilot. A quad-rotor model will be activated in simulation mode and feeds the measurement data into the *Mission planning* module. Meanwhile, the simulation data will be sent to the high-level autopilot via the *Serial comm*

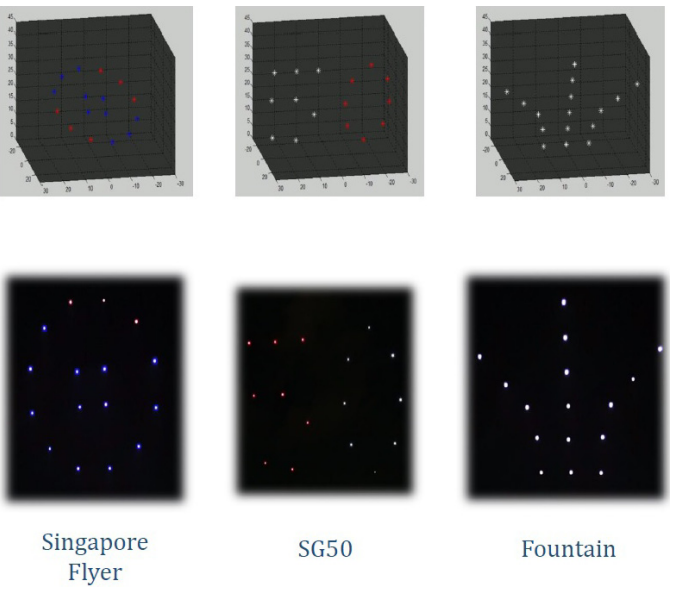

Fig. 18. MATLAB simulation performed for 16 UAVs.

module at 50 Hz. The hardware-in-the-loop simulation with all avionics components eases the development phase and intensive testing. The hardware-in-the-loop simulation will examine the onboard software system, GCS system, wireless communications, onboard path tracking, LED effects and visualize the performance effects of all the UAVs as shown in Fig. 19. After the hardware-in-the-loop simulation, a practical flight test can be performed with more confidence.

6.4. *Multiple-UAV flight strategy*

Given the formation flight requirements, the UAV team (16 UAVs) should take-off, start the predefined performance path and land synchronously. The synchronous behaviors of multiple-UAV is achieved with UTC received real-time from onboard GPS receivers. A flight plan for the multiple-UAV system is shown in Fig. 20. Once the UAV onboard system is powered on, the application will start automatically after boot up. Initially, the onboard program will be in $S0$: *Stand-by* state. In the *Stand-by* state, the UAV will perform all the tasks such as sensor data collection, data estimation, data logging, communication and so on. Once the pre-flight data collection and checking are normal, the ground pilot can issue the command to start the performance. The command "takeoff(h, t_1, n, t_2)" guides all the UAVs to take-off at UTC time of t_1 minute with a height of h meters. The n represents the onboard path identification number which will be performed at UTC time of t_2 minute. As shown in the state machine transition diagram, once onboard UTC minute time reaches t_1, the UAV will transit to $S1$: *Take-off* state and followed by the next behavior $S2$: *Fly to rendezvous point*

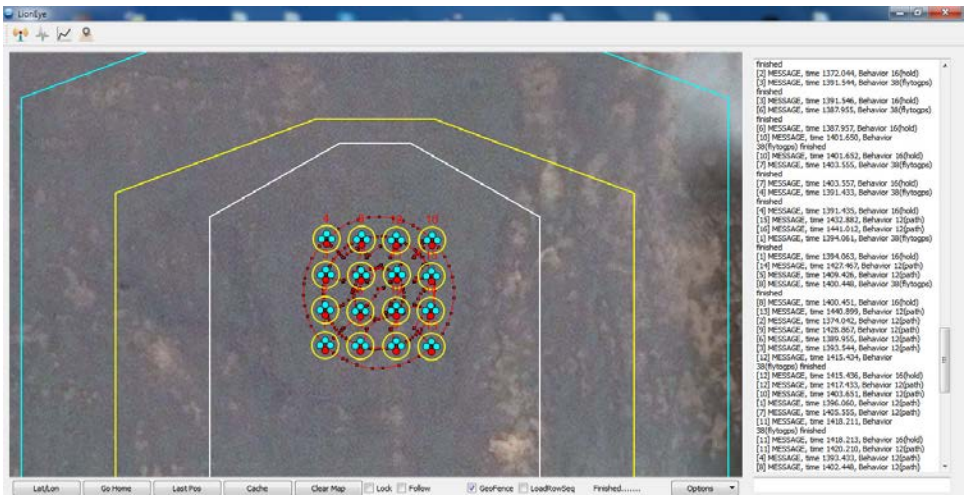

Fig. 19. Hardware-in-the-loop simulation result on GCS with 16 UAVs.

after the take-off height is reached. Before UTC time reaches t_2, the UAV will keep hovering before the performance begins. Then at UTC time minute of t_2, all the UAVs will start performing the path tracking behavior with predesigned formation patterns. Finally, all the UAVs will transit to state *S*4: *Return to home* after the performance is accomplished. All of the automatic behaviors are listed in Table 1.

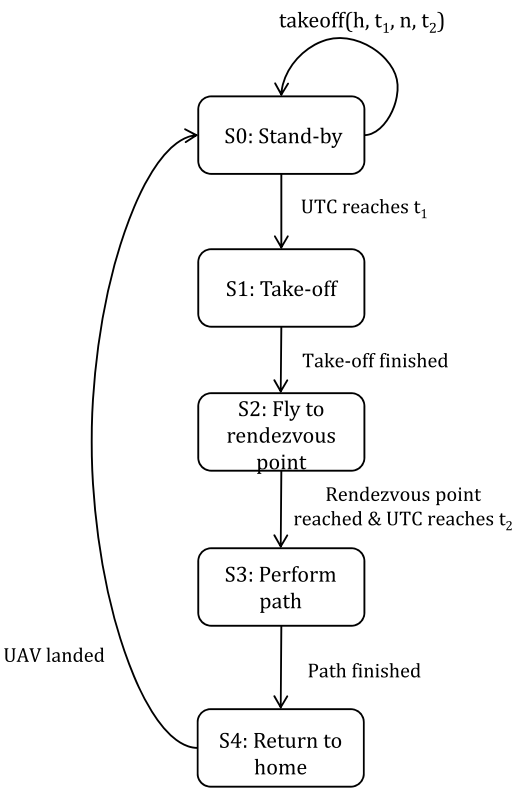

Fig. 20. Flight plan of UAV performance.

Table 1. Flight behaviors.

Flight behavior	Remarks
Low-level safety landing	Low-level autopilot automatic land.
Low-level safety descending	Low-level autopilot descend with only height control.
Low-level safety termination	Low-level autopilot cuts off motor.
High-level take-off	High-level autopilot generates take-off references.
High-level fly to rendezvous point	High-level autopilot guides each UAV.
High-level path tracking	High-level autopilot tracks the path reference.
High-level return home	Return to the take-off location and land.

Specifically, our scheduled performance time is set to start at 7:20 pm. Once the onboard application is started, our GCS can monitor the UTC time in hour, minute and second. During the *S*0: *Stand-by* state, the GCS user issues the command "takeoff(25, 19, 1, 20)". Then at 7:19 pm, all the UAVs will start to take-off to 25 m, then keep hovering until 7:20 pm when the onboard path will be performed.

7. Safety Features

From the very beginning of the design of both the software and hardware systems, safety features were taken into consideration and are always among the top priorities. In this section, a hierarchical three-layer safety architecture is

designed based on the risk assessment matrix which categorizes each flight failure condition based on the severity and probability of each case. Moreover, safety exit routes and a return procedure for multiple UAVs have been implemented to prevent them from mid-air collisions while flying out of their flight zone and to ensure their safe return to their launch location. The safety system has been implemented on 16 UAVs, formation flight demonstration and all the safety precautions have been verified in extensive flight tests.

7.1. *Software system safety implementation*

The software system running on the UAV onboard system is responsible for sensor data collection, sensor data processing, mission logic decision and flight control. All the functionality needs to be executed reliably in a real-time fashion. Considering the control structures and safety requirements for the drones, a two-layer software architecture is adopted for the UAV onboard system. One is the fundamental layer which controls the attitude of the UAV. The other higher layer is to send attitude references to the fundamental layer. With the combination of two-layer software structure, normal flight conditions and emergency situations are both covered.

7.1.1. *Flight safety design*

The safety precaution design for the multiple UAV formation is definitely a critical consideration. In case of hardware or software failure, a comprehensive approach to safety check and precautions is implemented given certain failure conditions as shown in Table 2. To accommodate system failures, a series of safety conditions are examined in both low-level and high-level autopilots. Once any safety condition is met in either autopilot system, the low-level autopilot on that particular UAV will take over the flight control to realize either landing or termination while other UAVs continue the flight performance. The safety precautions are not recoverable such that once a failure condition is met, the safety action will be executed until the UAV is landed. In case the precaution is not triggered in certain uncontrollable situations, the ground pilot still has the authority to switch to either semi-autonomous mode or manual mode.

Table 2. Flight safety design.

Causes of failures	Safety precautions
RC signal lost	Safety landing
High-level autopilot sensor reading stuck (invalid reading)	Safety landing
Low-level autopilot GPS no reading or invalid reading	Safety descending
1st and 2nd IMU on low-level autopilot fail	Flight termination
Soft geofence violation	Safety landing
Hard geofence violation	Flight termination
High-level autopilot software crash	Safety landing
Tracking error larger than threshold value	Safety landing
Onboard battery voltage low	Safety landing

7.2. *Hardware system safety implementation*

Our hardware safety system has four main inspections: pre-flight check, in-flight check, IMU online swap and real-time GCS health check monitoring. Pre-flight check, in-flight check and IMU online swap are integrated into the low level autopilot since it is more stable than the upper level autopilot. GCS health check monitoring is a GUI display feature for the operator to examine the real-time status of the system onboard.

7.2.1. *Pre-flight check*

Before flight, the UAV onboard system will check both the mechanical and avionic system separately. Firstly, when the system is armed, the four motors will spin in sequence at low speed to let the pilot confirm their functionality. Secondly, all the sensors' health situation including IMU and GPS will be checked before flight.

The program will start data processing after 60 sec of autopilot boot-up. This is to warm up all the onboard sensors include IMU and GPS. There are five kinds of data that are examined before UAV takes off:

(1) System Horizon: UAV horizontal inspection before take-off is very important, we need to ensure onboard autopilot IMU reading is correct instead of drifting and incline attitude reading due to calibration loss, EMI and improper UAV setup. In order to filter instant reading errors, program will capture 60 samples in a constant time interval, and calculate its mean value recursively. If the mean value is larger than a pre-defined threshold number, horizontal inspection is considered as a failure.
(2) Accelerometer: Accelerometer readings check is achieved by calculating the variances of three axis (X, Y, and Z) accelerometer readings.
(3) Magnetometer: Similar to the accelerometer reading, we consider magnetometer variance only. If any axis variance is larger than the thresholds, failure of inspection is triggered.
(4) GPS: GPS system in autopilot is a bit different from other sensors. The reading is not instantly published

when system boots up. The GPS module needs to wait for satellite lock. Normal waiting time is around one minute. Therefore, the program only starts to collect data when GPS is locked. We calculate GPS variances of horizontal position error (EPH) and vertical position error (EPV) value to judge the quality of GPS readings. EPH is a sigma horizontal error estimation in meters and EPV is a sigma vertical estimation error in meters.

(5) IMU stuck: There is a special case that may cause the above IMU sensor check to malfunction, and this happens when the IMU does not output a reading or is not powered. We recursively checked the current IMU readings and other previous four consecutive readings. When UAV is on ground, although the IMU readings are very small, they are not exactly zero. Hence, we use an experimental data threshold to judge what value relates to the IMU having no reading.

7.2.2. *In-flight check*

During the in-flight check situation, there are three emergency fail-safe actions involved which are "return home", "safety landing" and "flight termination".

- Return Home:
  When return home is activated, UAV will return to the launch site, each UAV will be defined a unique route for safe returning without collision. As shown in Fig. 21, return home action which is the least critical case, is possibly activated by crossing soft-geofence, remote radio link loss more than 10 sec, high-level autopilot IMU faults or no response.
- Safety Landing:
  Safety Landing is that UAV will land directly and slowly at current location and cut its throttle when the UAV has landed. This mode has the second highest priority, it will be triggered by the loss of the low-level auto-pilot GPS, battery critically low or UAVs in-air attitude angle exceed $\pm 45^{\circ}$. No GPS safety landing may cause UAV position drifting since the localization is purely based on onboard IMU and barometer.
- Flight Termination:
  Fight termination has the highest priority, it will be triggered when the low-level auto-pilot locks up or the hard-geofence is breached or several conditions happen together including one or both conditions in this mode.

7.2.3. *Avionics redundancy*

Failure of avionics would cause severe accident. So, we introduce avionics redundancy for key sensors, IMU and GPS.

IMU health condition directly decides the flight performance, our autopilot has a set of redundant IMU. If the primary IMU fails and results in no readings or reading stuck, the secondary IMU will stream its data into the attitude control loop.

It is difficult to judge whether the readings of IMU are correct or not when no reference data are available. The only way to check IMU condition is whether sensor reading is stuck. If reading variance for three consequence is zero, IMU is considered in failure status accordingly.

GPS provides the position information for the UAV, without which the UAV will not perform the position based task. GPS failure will cause the UAV to slowly drift to some

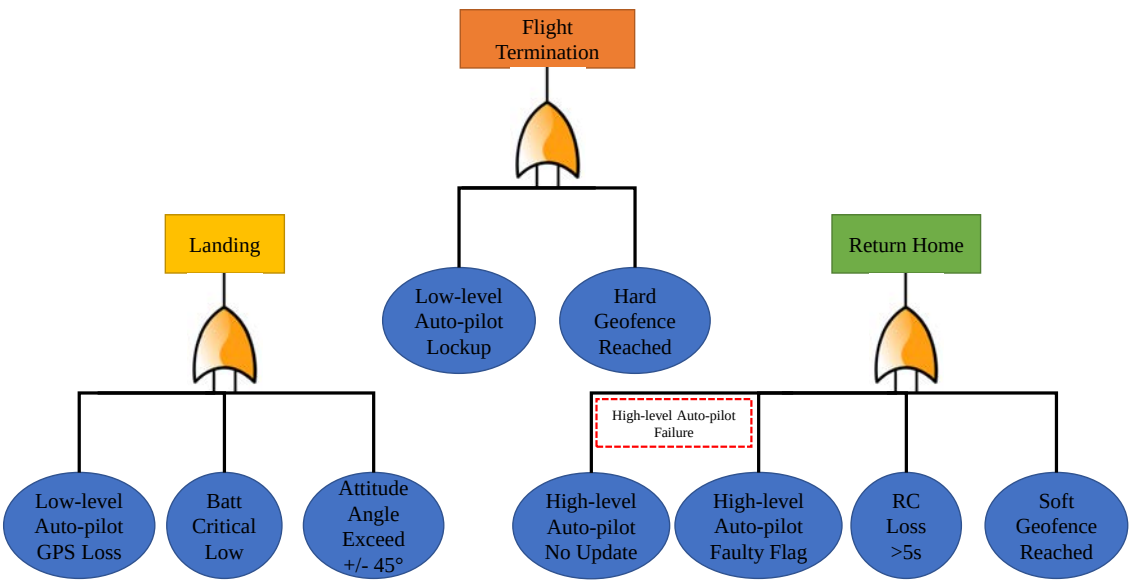

Fig. 21. Fault tree analysis diagram.

direction since the position feedback has not been updated. Besides the RTK GPS, we have also installed an uBlox GPS onboard. Whenever the RTK GPS fails, the uBlox GPS will instantly take over and the UAV will abort its mission and go into return home mode.

7.2.4. *EQT and ESS test*

Before finally assembling the UAV, all the avionics will go through environmental qualification test (EQT) and ESS test to verify their stability under extreme environment so that it can fit the outdoor usage. Besides, all the avionics will go through a water-proofing process such that the UAV will still function even under light rainfall. The purpose of ESS is to identify any product defects or manufacturing flaws by doing ful functionality test and ESS test. Its objectives are to remove manufacturing workmanship defects and eliminate infant mortality failures prior to usage.

7.2.5. *GCS health check monitoring*

To monitor the flight status of the UAV in real time, necessary data should be sent back to the GCS shown in Fig. 22, our setup is a semi-rugged Panasonic Toughbook. To ensure data is secured, we added a 128 bit Advanced Encryption Standard (AES) data encryption into the data link. The dataset was defined in both the air and ground side communication module. The GCS can monitor up to 32 UAV performance and show the key status of each UAV in one screen. Each UAV will occupy a single box and color code was used to represent the health status. When the box turns red, the UAV is diagnosed to be in unhealthy state. The operator can also broadcast commands to all UAV linked with GCS.

(1) Main Localization Data: GPS data with projected X, Y, Z coordinates and satellite number are shown in each UAV window. UAV headings based on high-level onboard compass is shown on the window as well.
(2) Battery Voltage: Battery Voltage is displayed on the screen, if voltage is lower than the preset threshold value, the onboard system will take failsafe action of safety landing.
(3) High-Level Flight Computer Modes: High-Level Flight Computer Modes are Take-off, Hold, Path, Land, Return Home and Idle. These modes are mainly used for formation path performance. In this paper, this part will not be expanded.
(4) Low-Level Autopilot Mode: This mode includes arm, geofence, failsafe, high-level faulty flag, battery condition, GPS and gyro reading flags. This information is integrated and encoded into 16 bit integer in order to decrease the data payload.
(5) Version Control: This includes low-level and high-level computer Firmware git commit number, a tag was

Fig. 22. GCS health monitoring window.

attached on the commit number when the repository is stable. GCS will detect firmware that is not in clean status, i.e. firmware has some changes after the stable version.

7.3. *Area safety implementation*

7.3.1. *Geofence system*

The performance center GPS coordinates are 1.2793553 N, 103.8622635 E. A program used for the auto-generation of three boundaries has been implemented. The program is fully based on the pre-designed shape of the boundary. If given previous GPS coordinate, using its bearing and distance to next boundary point, we could get the final geofence boundary coordinates to form a closed contour. We have set two geofence, the soft geofence and hard geofence to handle different faulty cases.

Performance Zone: Blue box in Fig. 23 is defined as the performance zone, the size of it is 80 m(L) × 80 m(W) × 60 m(H). This zone is the normal UAV operating zone during performance.

Soft Geofence: Green box is defined as the soft-geofence, the first protection of our area boundary. When the UAV reach the soft geofence, the UAV low-level computer will take over and guide UAV return home. To allow the show to operate smoothly without interruption if some failures happen, we defined a soft geofence boundary with a fixed size: 100 m(W) × 100 m(L) × 70 m(H). If the UAV low-level autopilot GPS detects that it crossed this boundary, it will be triggered as return to safety exit mode as shown in Fig. 22. UAV will return to launch site following the pre-defined unique route. To calculate the distance between the performance zone and soft geofence (20 m), we need to know what will happen when the UAV goes through the soft geofence boundary. The program handles geofence calculations at 0.1 s per loop. The geofence algorithm confirms geofence breach after 5 counts if out of geofence GPS coordinates exist. Therefore, if GPS detects UAV out of boundary, action will be triggered after 0.5 s. With the 2 m/s UAV ground speed and maximum 4 m/s wind speed (same direction with UAV), UAV will reach 3 m distance. Our UAV has maximum horizontal acceleration 3 m/s^2, we can calculate the maximum UAV drifting distance when it is out of boundary.

Hard Geofence: To ensure that the UAVs will never cross the safety boundary, a hard geofence was programmed. The UAV flight will be terminated upon crossing this geofence. This hard geofence is defined as 120 m(W) × 120 m(L) × 80 m(H). The max speed that the UAV could reach in the performance is 6 m/s (maximum speed) when the UAV velocity is directly perpendicular to the hard geofence boundary and its altitude is 80 m above sea level.

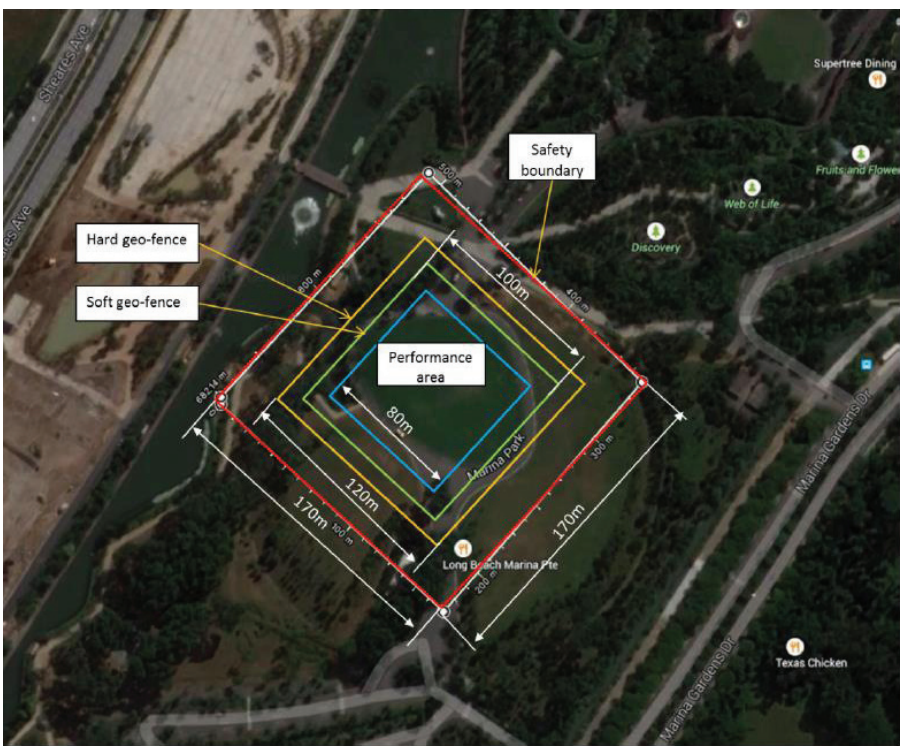

Fig. 23. Geofence design.

Fig. 24. Formation performance.

Safety Boundary: At the flying area, safety boundary is defined as a $170\,\mathrm{m(L)} \times 170\,\mathrm{m(W)} \times 80\,\mathrm{m(H)}$ geo-cubic. This is the worst case area which could be covered by the UAV which is shown in red color in Fig. 23.

7.4. *Flight test*

With the implementation of all the safety features explained in this paper, our system increases the multi-UAV control system's robustness and stability significantly. We eventually have the solution for very critical conditions such as IMU failure, GPS drifting or loss, etc. Figure 24 is the final result of our performance, we are able to fly up to 16 UAVs' formation safely.

8. Formation Flight Results

To verify the effectiveness and robustness of the proposed control scheme for the multiple UAVs, several flight experiments have been conducted with 16 UAVs. Selected UAV flight trajectories have been shown in Figs. 25–29. As shown in the figures, the UAVs can precisely track the predefined reference based on GPS synchronized time without communication between the UAVs and the GCS.

The experimental data shows that the proposed control scheme works well. The velocity response diagram in Fig. 27 shows UAV responds fast enough when reference changes so that it can catch up with the position reference. Since the formation performance is in an outdoor environment, wind disturbance applies to the UAV, thus the actual velocity in certain axis vary around 0 m/s when the velocity reference is zero. From Fig. 27, the velocity tracking error of the UAV is within 0.5 m/s, however, this error may vary based on the strength of the wind. Based on our developed precise velocity control, the UAV follows the trajectory of

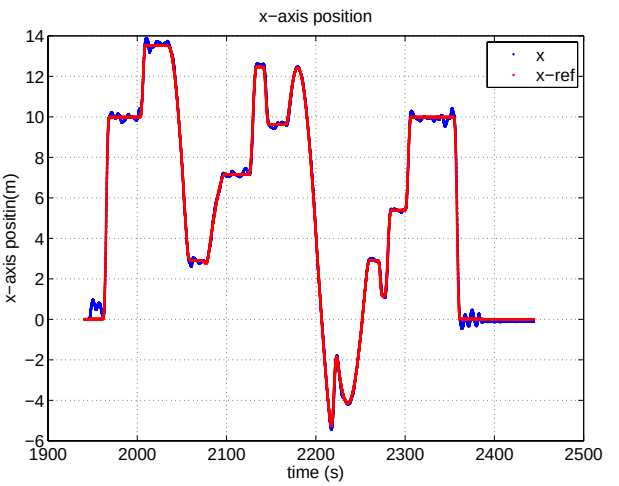

Fig. 25. UAV *x*-axis position flight performance.

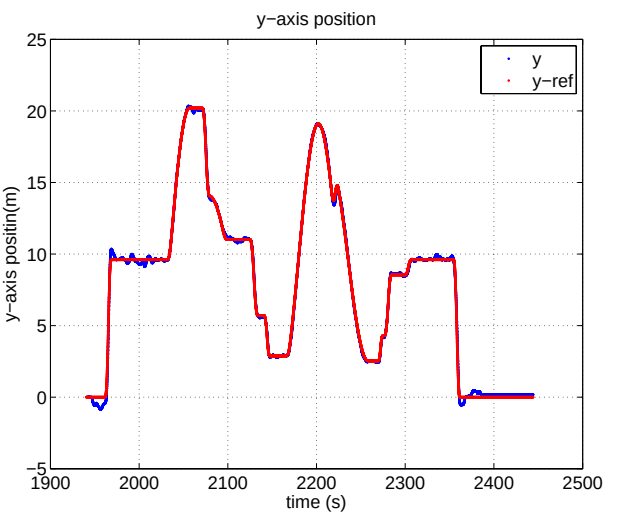

Fig. 26. UAV *y*-axis position flight performance.

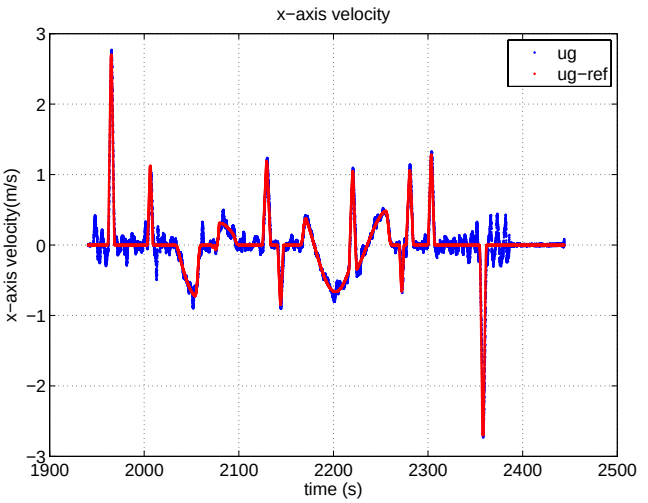

Fig. 27. UAV *x*-axis velocity flight performance.

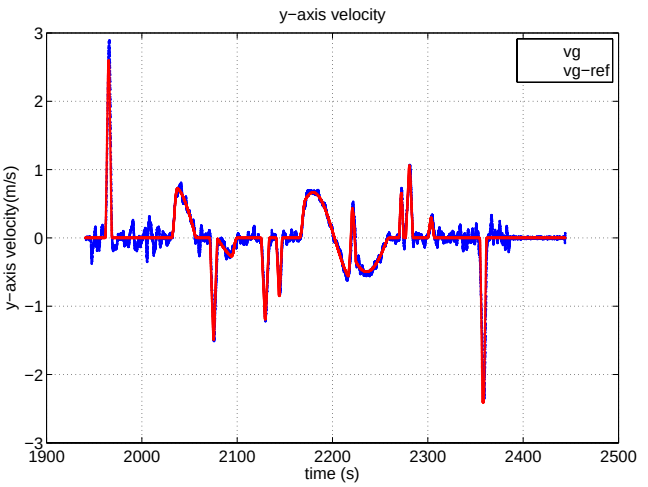

Fig. 28. UAV *y*-axis velocity flight performance.

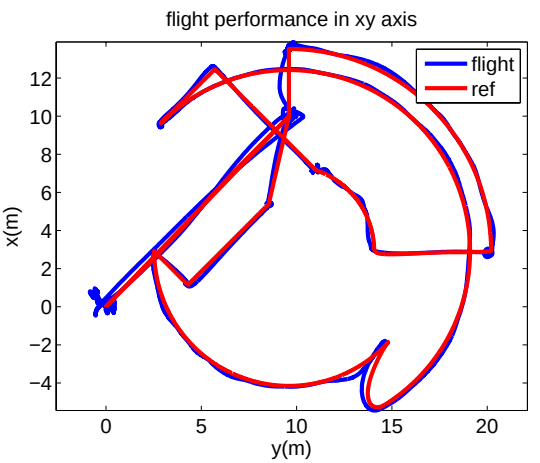

Fig. 29. Flight trajectory of a single UAV.

position reference very accurately. From Fig. 25, the maximum position tracking error is less than 1 m. The overshoot of the position control reaches a maximum of 0.45 m/s which is acceptable for our formation flight. Figure 29 shows the tracking performance in *x*–*y* plane. Our formation is composed of 16 UAVs with many interchanging formations, therefore, individual flight paths do not clearly indicate the patterns formed. However, it can still show the overall tracking performance of the UAV.

9. Conclusion and Future Work

In this paper, we have presented a complete documentation of the research work on technologies to realize precise formation and cooperation control of multiple UAVs. A system architecture has been proposed. To implement the proposed architecture, the UAV platforms have been successfully constructed, including the hardware platform and the cooperative software system. A comprehensive and accurate dynamical model of the UAV system has been identified and utilized to design a fully autonomous flight control law.

In addition, we have accomplished the modeling and controller design for the multiple UAV formation. The proposed UAV formation scheme has also been successfully verified in the flight tests under different configurations. The actual implementation results demonstrate that the formation fleet of the UAVs with our proposed formation control law is capable of completing a series of maneuvers accurately.

The robustness and performance of the proposed system has been tested in several flight demonstrations. These achievements of the system have been listed below:

- Developed a system framework suited for multiple UAV research and demonstration;
- Developed the synchronization scheme among UAVs based on GPS time;
- Developed the accurate positioning system and algorithms to fuse the RTK, DGPS, and IMUs;
- Developed the robust flight control system;
- Proposed and implemented sophisticated path planning and collision avoidance algorithms;
- Developed safety features for the UAVs, including hardware redundancy, BIT, in-flight check, reliability measures, and quality control, etc.
- Conducted performances in Gardens by the Bay.

Appendix A

The appendix shows the reference and trajectory for individual 16 UAVs.

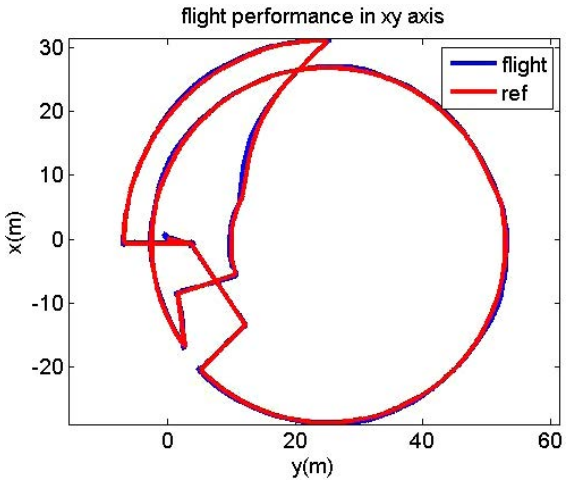

Fig. A.1. Flight trajectory of UAV1.

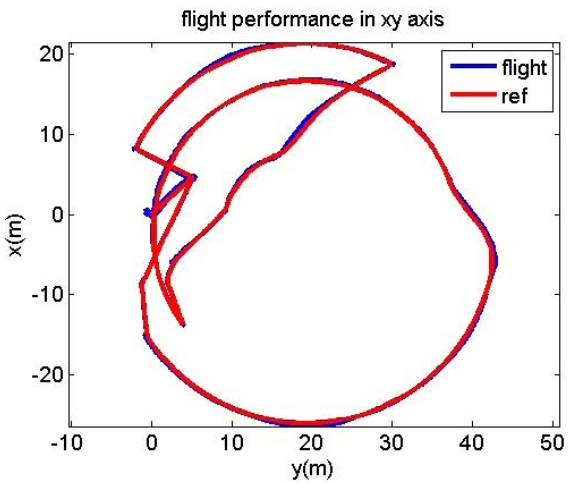

Fig. A.2. Flight trajectory of UAV2.

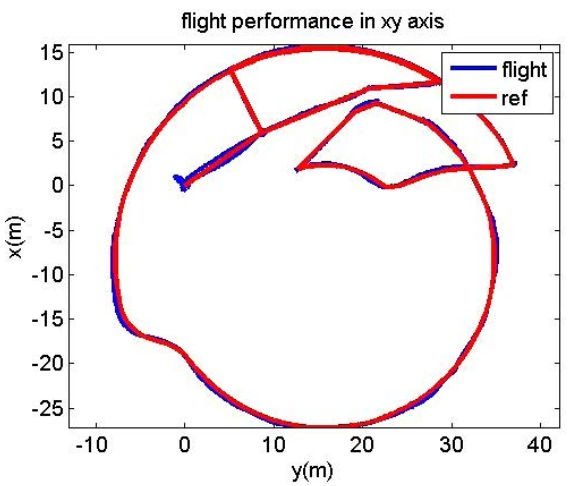

Fig. A.3. Flight trajectory of UAV3.

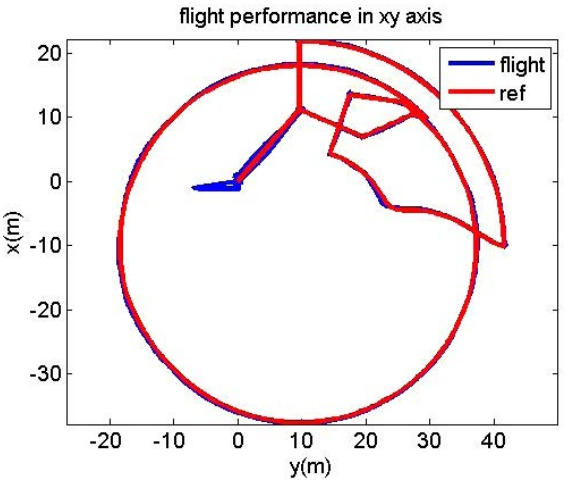

Fig. A.4. Flight trajectory of UAV4.

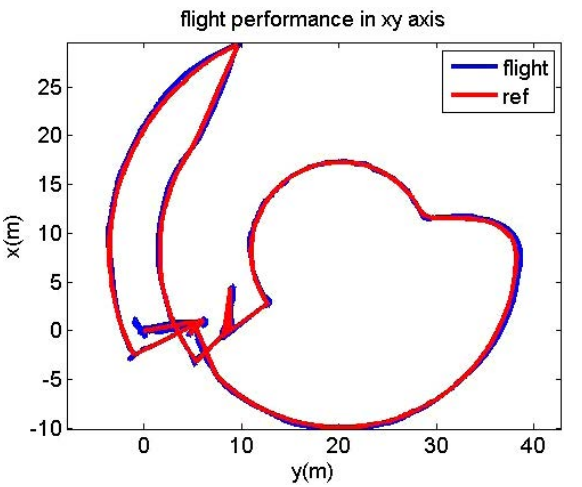

Fig. A.5. Flight trajectory of UAV5.

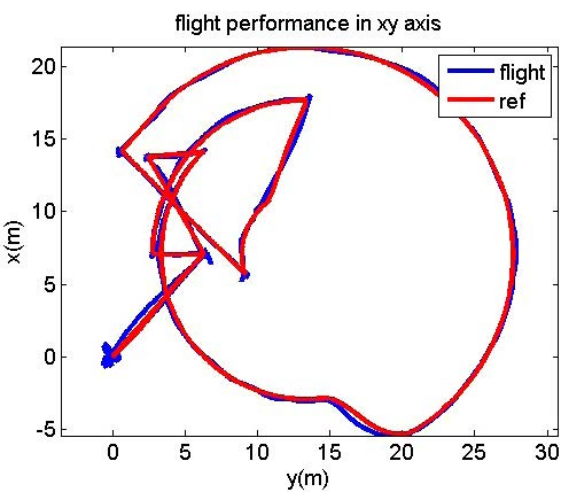

Fig. A.6. Flight trajectory of UAV6.

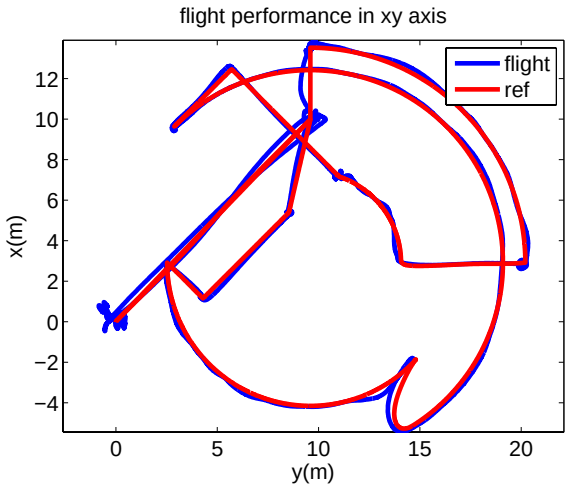

Fig. A.7. Flight trajectory of UAV7.

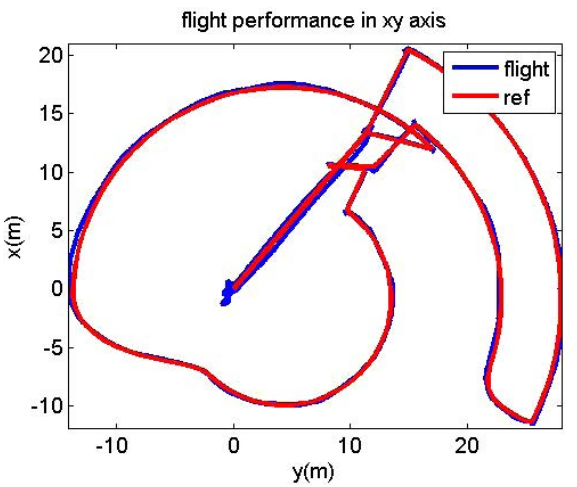

Fig. A.8. Flight trajectory of UAV8.

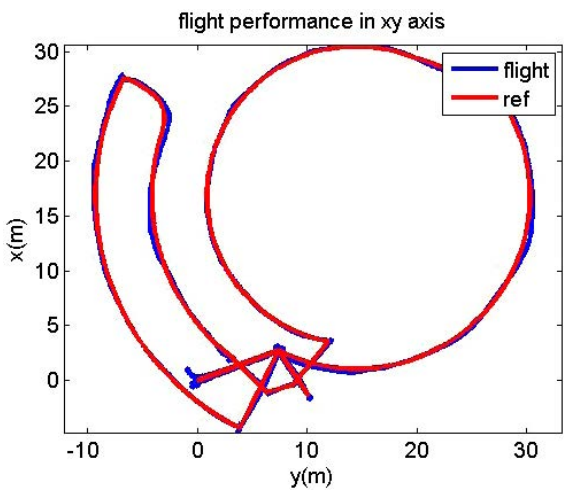

Fig. A.9. Flight trajectory of UAV9.

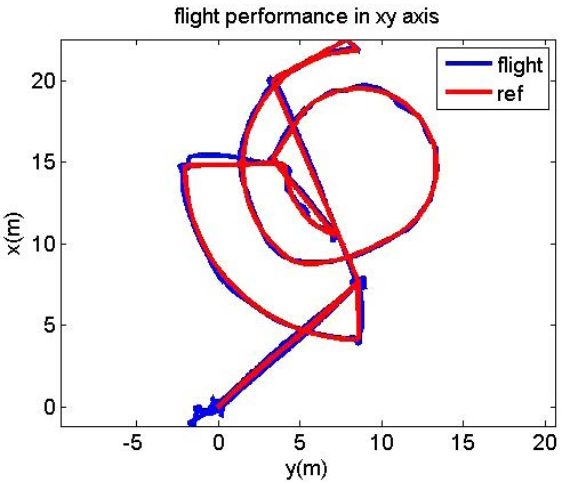

Fig. A.10. Flight trajectory of UAV10.

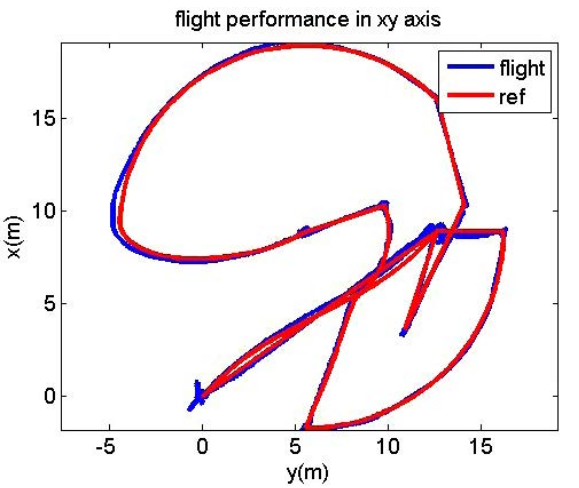

Fig. A.11. Flight trajectory of UAV11.

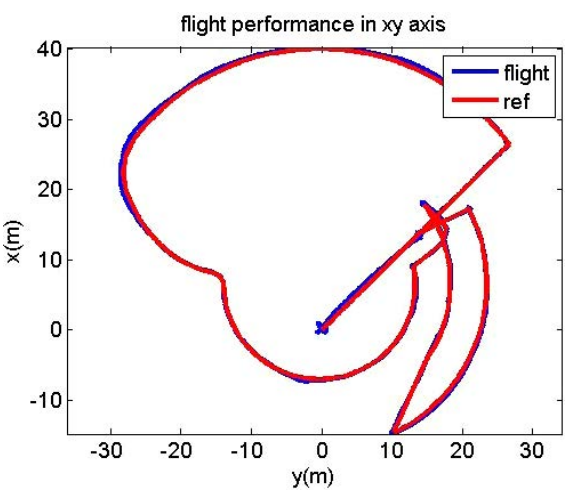

Fig. A.12. Flight trajectory of UAV12.

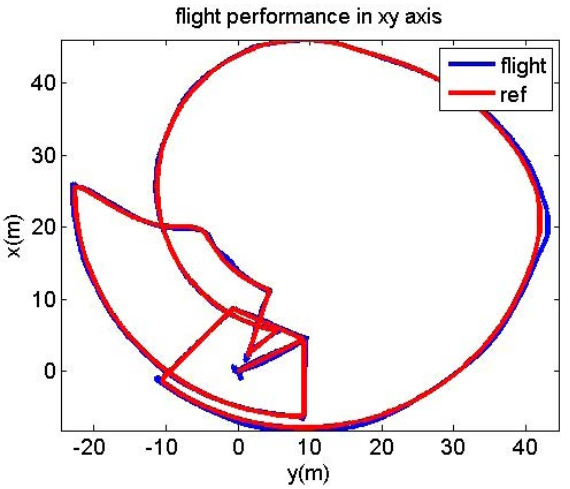

Fig. A.13. Flight trajectory of UAV13.

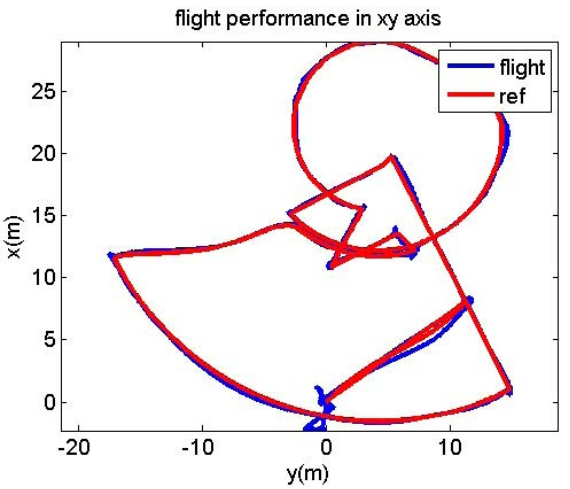

Fig. A.14. Flight trajectory of UAV14.

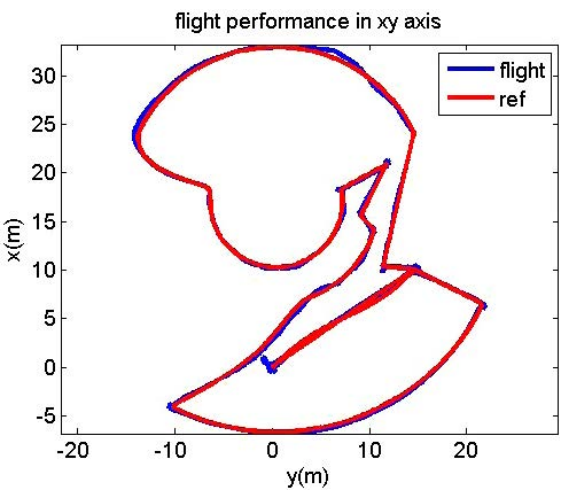

Fig. A.15. Flight trajectory of UAV15.

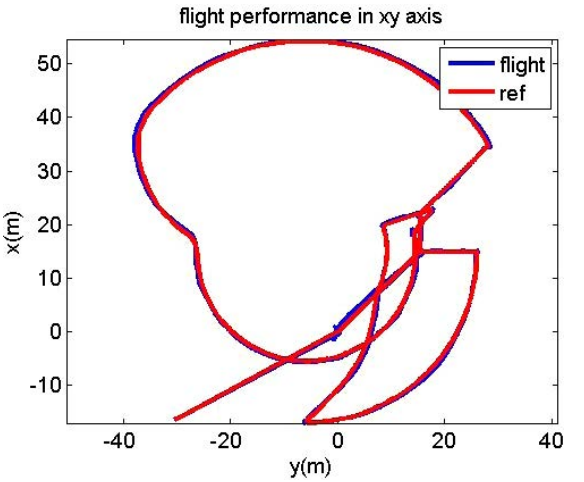

Fig. A.16. Flight trajectory of UAV16.

References

[1] X. Dong, Y. Zhou, Z. Ren and Y. Zhong, Time-varying formation tracking for second-order multi-agent systems subjected to switching topologies with application to quadrotor formation flying, *IEEE Trans. Ind. Electron.* **64**(6) (2017) 5014–5024.

[2] X. Dong, B. Yu, Z. Shi and Y. Zhong, Time-varying formation control for unmanned aerial vehicles: Theories and applications, *IEEE Trans. Control Syst. Technol.* **23**(1) (2015) 340–348.

[3] T. Shima and S. Rasmussen, *Uav Cooperative Decision and Control, Challenges and Practical Approaches* (SIAM, Philadelphia, 2009).

[4] X. Dong, Y. Zhou, Z. Ren and Y. Zhong, Time-varying formation control for unmanned aerial vehicles with switching interaction topologies, *Control Eng. Pract.* **46** (2016) 26–36.

[5] K. Peng, G. Cai, B. M. Chen, M. Dong, K. Y. Lum and T. H. Lee, Design and implementation of an autonomous flight control law for a uav helicopter, *Automatica* **45**(10) (2009) 2333–2338.

[6] F. Wang, K. Wang, S. Lai, S. K. Phang, B. M. Chen and T. H. Lee, An efficient uav navigation solution for confined but partially known indoor environments, in *11th IEEE Int. Conf. Control & Automation (ICCA)* (IEEE, 2014), pp. 1351–1356.

[7] S. Lai, K. Wang, H. Qin, J. Q. Cui and B. M. Chen, A robust online path planning approach in cluttered environments for micro rotorcraft drones, *Control Theory Technol.* **14**(1) (2016) 83–96.

[8] F. Lin, K. Z. Ang, F. Wang, B. M. Chen, T. H. Lee, B. Yang, M. Dong, X. Dong, J. Cui, S. K. Phang *et al.*, Development of an unmanned coaxial rotorcraft for the darpa uavforge challenge, *Unmanned Syst.* **1**(2) (2013) 211–245.

[9] K. Z. Ang, J. Cui, T. Pang, K. Li, K. Wang, Y. Ke and B. M. Chen, Development of an unmanned tail-sitter with reconfigurable wings: U-lion, in *11th IEEE Int. Conf. Control & Automation (ICCA)* (IEEE, 2014), pp. 750–755.

[10] B. M. Chen, T. H. Lee, K. Peng and V. Venkataramanan, *Hard Disk Drive Servo Systems* (Springer Science & Business Media, 2006).

[11] B. M. Chen, *Robust and H∞ Control* (Springer Science & Business Media, 2013).

[12] J. Q. Cui, S. Lai, X. Dong and B. M. Chen, Autonomous navigation of uav in foliage environment, *J. Intel. Robotic Syst.* **84** (2015) 259–276.

[13] S. K. Phang, S. Lai, F. Wang, M. Lan and B. M. Chen, Systems design and implementation with jerk-optimized trajectory generation for uav calligraphy, *Mechatronics* **30** (2015) 65–75.

[14] T. Kröger, Opening the door to new sensor-based robot applications: The reflexxes motion libraries, in *2011 IEEE Int. Conf. Robotics and Automation*, May 2011, pp. 1–4.
[15] H. Kano, H. Nakata and C. F. Martin, Optimal curve fitting and smoothing using normalized uniform b-splines: A tool for studying complex systems, *Appl. Math. Comput.* **169**(1) (2005) 96–128.
[16] H. Kano, H. Fujioka and C. F. Martin, Optimal smoothing and interpolating splines with constraints, *Appl. Math. Comput.* **218**(5) (2011) 1831–1844.
[17] H. Fujioka and H. Kano, Control theoretic B-spline smoothing with constraints on derivatives, in *52nd IEEE Conf. Decision and Control*, December 2013, pp. 2115–2120.
[18] T. Kröger and F. M. Wahl, Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events, *IEEE Trans. Robotics* **26**(1) (2010) 94–111.
[19] R. Haschke, E. Weitnauer and H. Ritter, On-line planning of time-optimal, jerk-limited trajectories, in *2008 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, September 2008, pp. 3248–3253.
[20] T. Kröger, Online trajectory generation: Straight-line trajectories, *IEEE Trans. Robotics* **27**(5) (2011) 1010–1016.
[21] J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Indus. Appl. Math.* **5**(1) (1957) 32–38.