International Journal of Robotics and Automation, Vol. 26, No. 1, 2011

DEVELOPMENT OF A COMPREHENSIVE SOFTWARE SYSTEM FOR IMPLEMENTING COOPERATIVE CONTROL OF MULTIPLE UNMANNED AERIAL VEHICLES

Xiangxu Dong,* Ben M. Chen,* Guowei Cai,* Hai Lin,* and Tong H. Lee*

Abstract

In this work, we focus on establishing a framework and developing a comprehensive software platform, which is capable of realizing the cooperative control for multiple unmanned aerial vehicles (UAVs). An efficient and flexible framework is adopted for cooperative control law design, on which we build up a software platform consisting of an onboard real-time software system for UAVs and a ground control station (GCS). Such a platform employs a distributed architecture to facilitate task deployment, efficient monitoring and commanding the UAVs via the ground station. Hardware-in-the-loop simulation and practical cooperative flight experiments have been conducted to verify the efficiency of the overall software system.

Key Words

Unmanned aerial vehicle, cooperative control, formation flight, real-time systems, software systems

1. Introduction

The prevalent military and civilian deployment of unmanned vehicles has been clearly seen in the last two to three decades. Unmanned vehicles can outperform humans in many applications and protect humans efficiently in various dangerous situations. Working as a group, for unmanned systems, will greatly enhance their efficiency and introduce new working paradigms in the unmanned systems. As such, research in the field of control and coordination for multiple unmanned autonomous vehicles has aroused great interest recently. Representative topics include universal flight control system design [1], localization [2], mapping and exploration [3, 4], surveillance [5], search and rescue [6], object transportation and manipulation [7, 8].

With the recent technology boosting in sensor, communication and computation, various components which are necessary for onboard computer system construction have become increasingly smaller and more powerful than before. However, there are still many challenging problems in the cooperative unmannied aerial vehicle (UAV) control [29]. As such, many platforms for demonstrations of the cooperative control of multi-vehicles have been developed worldwide. For instance, the research group in MIT [10, 11] has demonstrated the formation flight with the receding horizon framework. The researchers in Stanford University [12] have used the quad rotor helicopters as their testbed STARMAC. At University of Pennsylvania [13], a hybrid system approach is used for the cooperative control among multiple fixed-wing UAVs. The work of [23] has managed to exchange information with an aircraft-to-aircraft ad-hoc wireless network. At Brigham Young University, the MAGICC UAVs [14] are developed for the multiple-vehicle research. In all, a reliable, distributed and scalable multiple-UAV platform is preferred [15].

In this paper, the concentration is on the control and coordination of a group of UAVs. More specifically, we focus on the formation flight of multiple UAVs. Taking our single-UAV-based software system documented in [16] as the baseline, we in this work aim to establish an efficient framework for multiple-UAV coordination control and build up a comprehensive software system to realize the reliable coordination in practical flight tests, based on our self-constructed Raptor 90 helicopter platforms, namely HeLion and SheLion (see Fig. 1).

The outline of this paper is as follows. First, we introduce the framework for coordinate control of multiple UAVs and explain the approach of performing formation flight cooperative control under such a framework in Section 2. In Section 3, the software system for realizing the UAV coordination control is presented. Its two key components, the onboard software and ground control station, are both addressed. In Section 4, we present the hardware-in-the-loop simulation and practical flight test results and detailed analysis of the coordination flight. Finally, we draw some conclusion remarks in Section 5.

* Department of Electrical & Computer Engineering, National University of Singapore, Singapore 117576; e-mail: {dong07, bmchen, tslcaig, elelh, eleleeth}@nus.edu.sg

Recommended by Prof. Z. Qu
(10.2316/Journal.206.2011.1.206-3417)

Figure 1. Raptor 90 helicopter, HeLion.

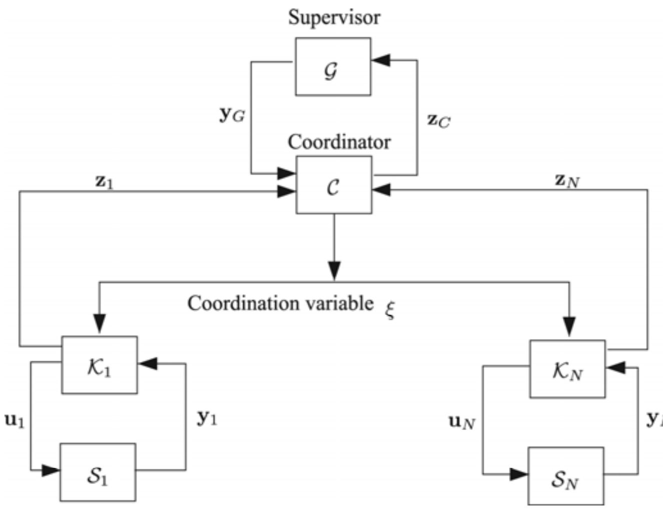


Figure 3. Architecture for coordinated control among multiple UAVs.

2. Framework of Coordination Control

In this section, we propose a modular framework for cooperative control of the multiple-UAV system for the formation flight. A typical scenario of formation flight is shown in Fig. 2, which consists of multiple UAVs and a solo ground control station. Each UAV is required to communicate with other UAVs and interact with the GCS simultaneously. To guarantee the reliability of formation flight, a simple but efficient cooperative control framework is compulsory. In what follows, we first introduce the general UAV coordination control architecture, and then zoom into the details of one specific implementation, i.e., the leader–follower based formation flight.

2.1 General Architecture for Coordination Control

The general architecture of the coordination control for multiple UAVs is depicted in Fig. 3. It is organized in hierarchical layers to accommodate practical requirements. It consists of three layers and the highest layer coordinates the dynamic transitions from one state to another in the overall coordination task. The idea of the adopted architecture for our application-oriented project comes from [17]. The detailed description for these three layers are presented as follows:

1. The lowest layer focuses on each single UAV unit. Specifically, $\mathcal{S}$ represents the dynamic model of the individual UAV, with the control input vector $\mathbf{u}_i$ and the measurable output vector $\mathbf{y}_i$, and $\mathcal{K}$ is the representation of the local controller for the UAV. It should be noted that the inputs to $\mathcal{K}$ are the coordination variable ξ and output $\mathbf{y}_i$ and the outputs of $\mathcal{K}$ are the coordinate performance variable $\mathbf{z}_i$. The coordinate performance may have different definitions according to the coordinate behaviours, such as formation flight we address later.

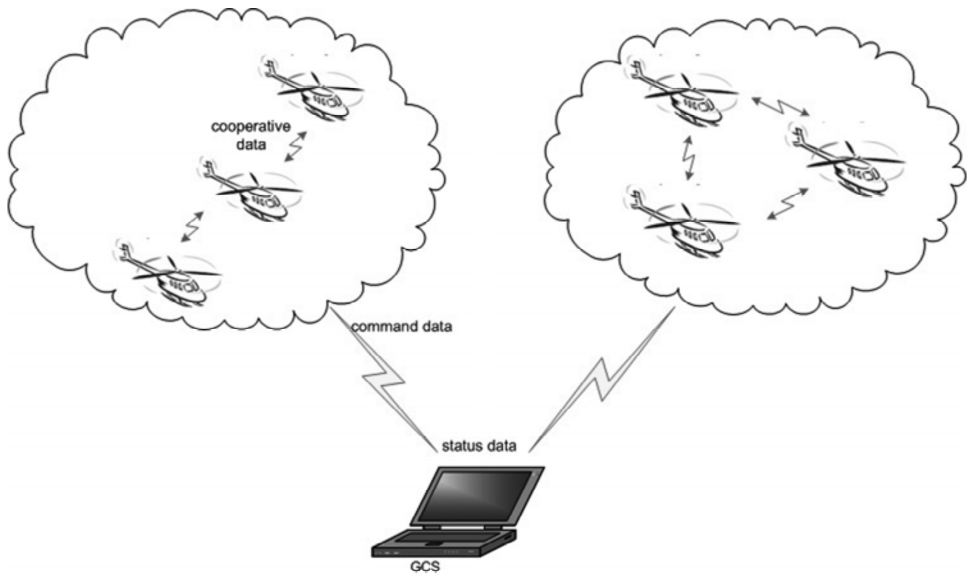


Figure 2. Formation flight scenario.

2. The middle layer of the overall architecture is the coordinator $\mathcal{C}$. It receives coordinate performance input vector from some (or all) UAVs, then processes and encapsulates the performance evaluation result vector z_C to the top layer according to the coordination mechanism. It adjusts the coordination mechanism based on the performance feedback $\mathbf{y}_G$ from top layer. The output of coordinator ξ_i behaves as the interaction between the local UAV and the global team and can be broadcasted or multicasted to the UAV team.
3. $\mathcal{G}$, which is a discrete-event system, is located at the highest level. It acts as a supervisor to regulate the performance of coordination for the multiple-UAV system. It receives the performance vector from the coordinator and generates the inputs $\mathbf{y}_G$ for the coordinator.

It is clear to see that such generalized coordination control architecture is suitable for various control strategies. For example, the lowest layer can hire different flight control laws to realize automatic control for a single UAV; the coordinator can choose different coordination mechanism based on the specific coordination behaviours such as rendezvous, collision avoidance, and dispersion. The coordination mechanism determines how the subtasks are correctly transferred from one subtask to another in the perspective of system overall behaviours. In summary, the presented coordination architecture proposed in Fig. 3 is a flexible and modular framework suitable for exploring real-time cooperative behaviours. Next, we focus on a specific coordination behaviour, formation flight of multiple UAVs.

2.2 Formation Flight

With the general architecture in hand, we here proceed to carry out study on a specific case: formation flight for multiple UAV helicopters. Leader–follower approach is adopted in our work and its general scheme is illustrated in Fig. 4. x_g, y_g, and z_g represent the North-East-Down frame. x_w, y_w, and z_w represent the wind frame associated with each individual UAV. The position of the leader is denoted by the point L, while the points D and A denote the desired position and actual position of the follower, respectively. The distance vector $\vec{R}_{LW}$, derived from the difference between $\vec{R}_W$ and $\vec{R}_L$, denotes the distance offset between the follower and the leader. We define the formation requirements such that f_c, l_c, and h_c meet the distance offset specifications. f_c, l_c, and h_c represent the longitudinal, lateral and vertical distance offset between the leader and the follower. In this scenario, the coordination is achieved through the leader UAV and coordination data is from the leader, with $\xi = \mathbf{x}_i$. For this specific case, we implement the above mentioned coordination control architecture as follows.

2.2.1 UAV Helicopters

Two UAV helicopters, namely HeLion and SheLion, are adopted as the platform. With the onboard computer stacks equipped, we have successfully realized stable

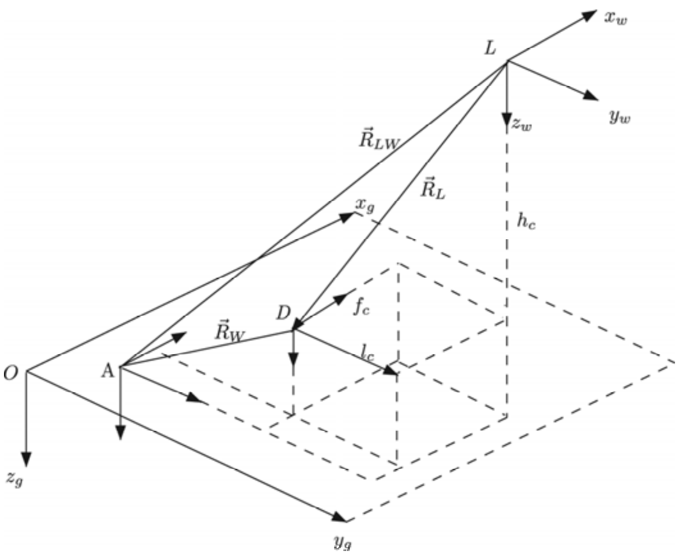


Figure 4. Leader–follower formation geometry.

automatic flight tasks such as vertical takeoff, hover, path tracking and landing for each of single UAV helicopter.

The overall control architecture is shown in Fig. 5. It consists of two parts: the outer-loop controller and the inner-loop controller. The inner-loop is to realize stable attitude control and outer-loop is to provide reference signals for inner-loop to realize trajectory tracking. Therefore, for the control of formation flight, the outer-loop can be regarded as a reference path generator. As such, the outer-loop serves as the interface between the team coordination task and local task. The dispatched task from the coordinator is executed by the outer-loop. The task of the inner-loop controller is to follow the output of the outer-loop controller. Interested readers on the controller design details are referred to [18].

2.2.2 Coordinator

For the coordinator in formation flight, we define the the formation state vector of the overall UAV group as

$$\mathbf{x}_F = [\mathbf{x}_{F1}, \mathbf{x}_{F2}, \ldots, \mathbf{x}_{Fn}]^T \tag{1}$$

where $\mathbf{x_{Fi}}$ is the state vector for the ith UAV and given by

$$\mathbf{x}_{Fi} = [x, y, z, c]^T \tag{2}$$

Here x, y, z, and c represent the longitudinal position, lateral position, altitude and heading angle in north-east-down (NED) frame, respectively.

Performance indices are further defined. Specifically, we first consider the constraint on collision avoidance, which is critically important to ensure the safety of the UAV group. For our UAV helicopter platforms, the Euclidean distance between two adjacent UAV helicopters, $\mathbf{E_a}(\mathbf{x}_{Fi}, \mathbf{x}_{Fj})$, is first adopted. The second important performance index is the heading angle consistency of the UAV members, which is denoted as $\mathbf{A}(\mathbf{x}_{Fi}, \mathbf{x}_{Fj})$. Finally, the Euclidean distance between the current position and reference position for the ith UAV, that is, $\mathbf{E_b}(\mathbf{x}_{Fi}, \mathbf{x}^r_{Fi})$, is also

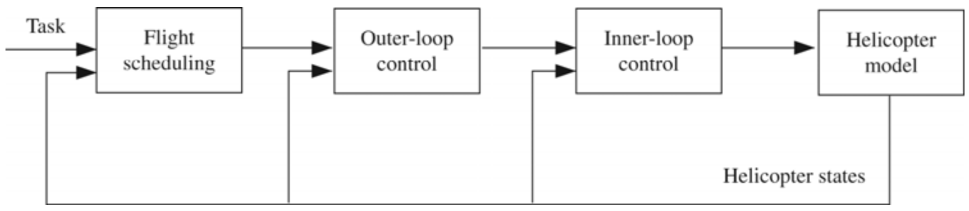


Figure 5. The structure of the UAV control system.

taken into account. We then summarize the performance triplet for each UAV as

$$\begin{cases} E_{ai}(\mathbf{x}_{Fi}, \mathbf{x}_{Fj}) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \\ E_{bi}(\mathbf{x}_{Fi}, \mathbf{x}_{Fi}^r) = \sqrt{(x_i - x_i^r)^2 + (y_i - y_i^r)^2 + (z_i - z_i^r)^2} \\ A_i(\mathbf{x}_{Fi}, \mathbf{x}_{Fj}) = |c_i - c_j| \end{cases} \tag{3}$$

where $\mathbf{x}_{Fi}^r$ is the reference position for the ith UAV. Our current implementation case involves only two UAV helicopters, UAV$_2$ is the leader while UAV$_1$ is the follower. In this case, the performance indices encapsulated $\mathbf{z}_C$ for the coordinator are modified as

$$\mathbf{z}_C : \begin{cases} E_a = E_{a1} \\ \mathbf{E}_b = [E_{b1}, E_{b2}]^T \\ A = A_1 \end{cases} \tag{4}$$

After determining the performance vector $\mathbf{z}_C$, the coordinator sends it to the supervisor for state transitions.

2.2.3 Supervisor

The supervisor receives the updated performance vector from the coordinator and conducts state transitions based on the formation flight requirements such as the tolerable Euclidean distance, time limitation for reaching certain reference position(s).

Supervisor is first in the formation initialization state, in which the supervisor drives the coordinator to send rendezvous command to each UAV helicopter. Once the updated formation performance indicates that both leader and follower have reached their desired positions, the supervisor switches to the next state: dispatching the task of performing formation flight. In this state, collision avoidance is periodically examined based on the updated performance index. If the threshold condition of collision is satisfied, the supervisor will command all of the UAV helicopters to hover at their current positions. Once the collision alarm disappears, the supervisor will send command to resume the formation flight. The formation flight task is completed till no performance update (when the follower receives no more update references) is received from the coordinator. Fig. 6 illustrates the state transition diagram in the supervisor when performing a leader–follower formation flight. The conditions are listed in Table 1,

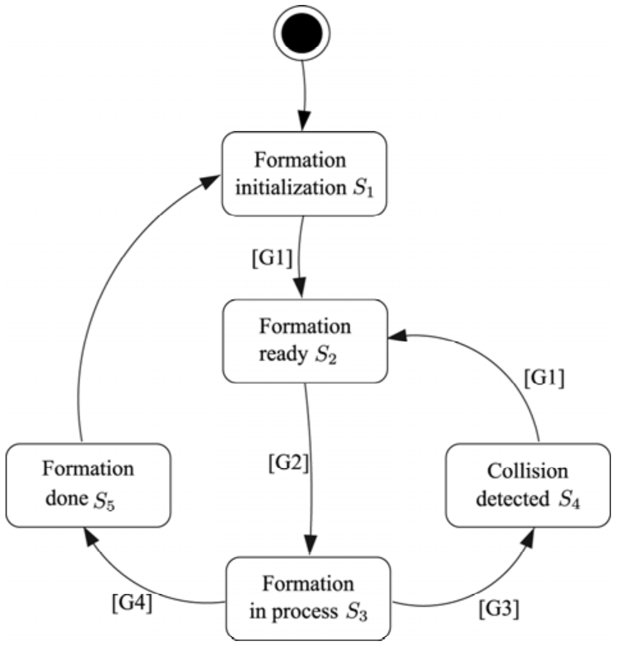


Figure 6. State transition diagram in supervisor $\mathcal{G}$.

Table 1
Performance Evaluations in Supervisor $\mathcal{G}$

Performance	True	False
G1	$\mathbf{E}_b > \epsilon_1$	Otherwise
G2	$\epsilon 2 < \mathbf{E}_a < \epsilon_3$ and $\alpha_1 < \mathbf{A} < \alpha_2$	Otherwise
G3	$\mathbf{E}_a < \epsilon_4$	Otherwise
G4	$\mathbf{z}_C(t) = \mathbf{z}_C$	Otherwise

where ϵ_1 is the tolerable distance difference in hovering state, ϵ_2 is the tolerance of distance between leader and follower, ϵ_3 is the tolerance (largest) distance for detecting the collision avoidance and α_1 and α_2 are the lower and upper boundaries of the difference in the heading angle.

The supervisor output $\mathbf{y}_G$ to the coordinator includes the tasks for each UAV helicopters. For formation flight, the leader and follower are both assigned the path tracking task. If collision avoidance is required, both the leader and follower UAV helicopters are commanded to perform hovering. Table 2 lists the output of the supervisor in different states. The output $\mathbf{y}_G$ is the coordination vector consisting of the ID of the UAVs involved in the current task, assigned task for each UAV and task parameters. The first row of $\mathbf{y}_G$ in Table 2 specifies UAV$_1$ and UAV$_2$

Table 2
Supervisor Output $\mathbf{y}_G$

States(S_i)	Output($\mathbf{y}_G$)
S_1	($i=1$, $j=2$, BEHAVIOR_HEADTO, BEHAVIOR_HEADTO, $\mathbf{x}^r_{Fi}$, $\mathbf{x}^r_{Fj}$)
S_2	($i=1$, $j=2$, BEHAVIOR_HOLD, BEHAVIOR_HOLD, $\mathbf{x}^r_{Fi}$, $\mathbf{x}^r_{Fj}$)
S_3	($i=1$, $j=2$, BEHAVIOR_PATH, BEHAVIOR_PATH, $\mathbf{x}^r_{Fi}(t)$, $\mathbf{x}^r_{Fj}(t)$)
S_4	($i=1$, $j=2$, BEHAVIOR_HOLD, BEHAVIOR_HOLD, $\mathbf{x}^r_{Fi}$, $\mathbf{x}^r_{Fj}$)
S_5	($i=1$, $j=2$, BEHAVIOR_HOLD, BEHAVIOR_HOLD, $\mathbf{x}^r_{Fi}$, $\mathbf{x}^r_{Fj}$)

to perform "headto" commands, and $\mathbf{x}^r_{Fi}$ and $\mathbf{x}^r_{Fj}$ specifies the reference position and heading angle that they should follow.

3. Software System

To realize the coordination control in practical implementations, a comprehensive software system has been developed. It mainly consists of two parts: the onboard software and the ground control station. In what follows of this section, we address both of them in detail.

3.1 Onboard Software

The onboard software is in charge of collecting information from the avionic sensors, executing the algorithms of coordination control, driving the servo actuators, exchanging information with other UAVs and transmitting data to the GCS for monitoring purpose. We first introduce the tasks and the overall architecture of the onboard software, and then present the schemes for task management and control law implementation.

3.1.1 Tasks and Architecture

For the onboard software, multiple tasks (threads) architecture is adopted. Among these tasks, one task is particularly assigned to act as supervisor and coordinator for coordination control purpose, and the remaining tasks are related to the single-UAV components such as wireless serial communication (CMM), WiFi card, inertial measurement unit (IMU), data acquisition (DAQ) board, and servo actuators. Such architecture is also hierarchical and consists of five layers, which is shown in Fig. 7.

The communication module realizes the information exchange among UAVs and data feedback to GCS. In our onboard software, the information exchange method among UAVs is selectable: either centralized or decentralized. Furthermore, the communication block is in charge of receiving ground user's commands to perform new flight missions. The commands received by the communication block are parsed and dispatched into the corresponding behaviours and flight control law.

The onboard data flow is contributed by various avionic sensors such as IMU and DAQ board. The measurable output includes position, velocity, attitude from IMU, main rotor RPM (rotations per minute), and sonar-measured

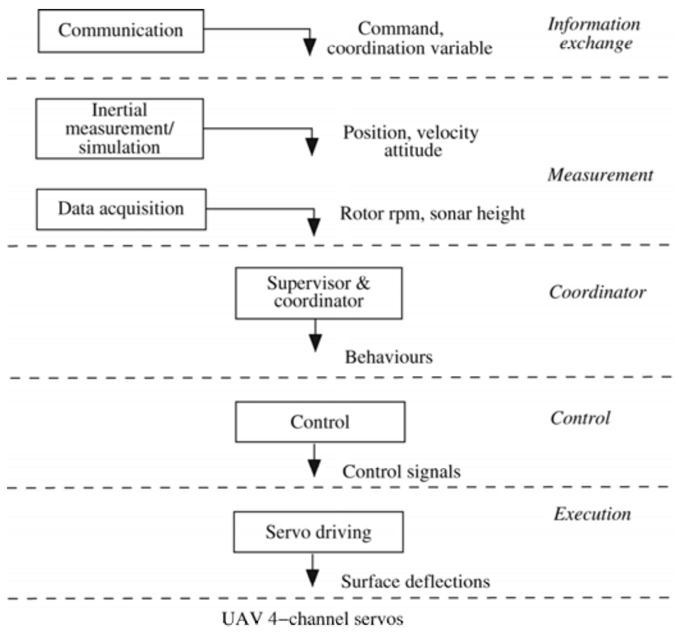


Figure 7. Onboard software architecture.

height values. The primary reason for including sonar-measured height is the relative large altitude error measured by GPS. It should be noted that, when conducting the ground simulation, a special simulation block is utilized to virtually generate the IMU measurement output.

The supervisor and coordinator are employed to combine the data received from the communication block (such as coordination variable and flight status of other UAVs) and its own status to derive the coordinate behaviour. For example, in a centralized form of formation flight, the supervisor in the leader determines the state transition based on the input from coordinator. For the followers, their formation flight status data will be sent to the coordinator in the leader.

With the updated status and dispatched coordinate behaviour, the control block will be activated to derive the control signal output for the execution block based on the well-designed control law.

The execution block refers to the servo actuator driving. It feeds the output of control block into individual servo actuators (for helicopter, including aileron, elevator, auxiliary, and rudder) to drive the surface deflections to the desired positions.

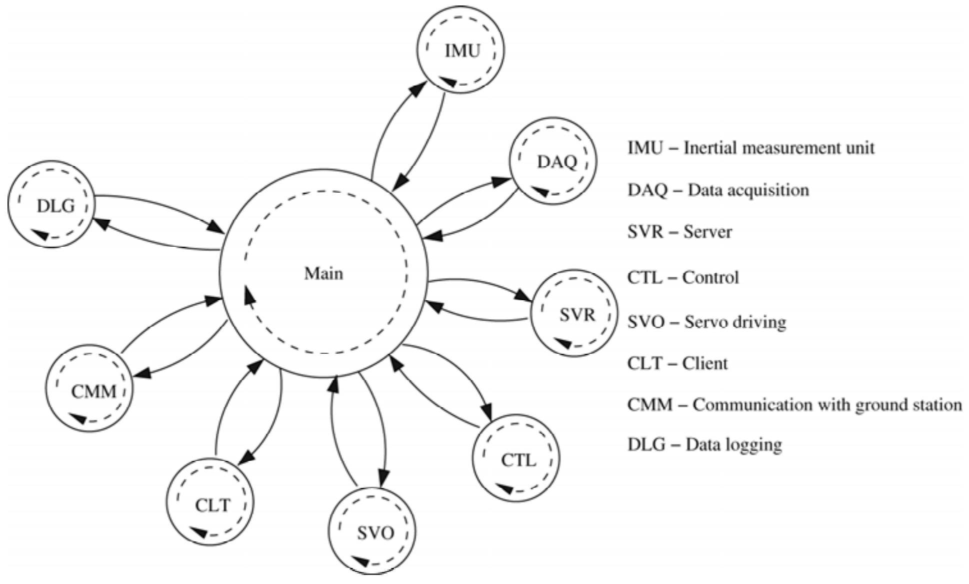


Figure 8. Onboard tasks management.

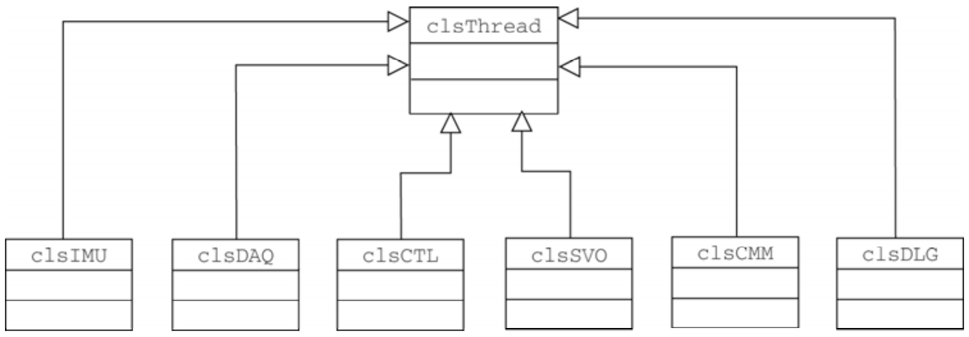


Figure 9. Onboard class diagram.

In our previous design, ground station communicates with UAVs *via* the wireless transceiver with 115.2 Kbps transmission rate. Such communication throughput is sufficient for multiple-UAV formation flight. Meanwhile we hire a new wireless communication protocol, that is, 802.11b, to handle the coordination control scenarios in which larger data throughput is required. In our current applications, the low data rate modem link is used for long range communication (flight data of each helicopter is returned to ground station), while 802.11b is used for relative short range and high data rate communication, such as information exchange for cooperation and computing. The overall working principle of the architecture is as follows: each UAV periodically transmits its data packet to GCS *via* the wireless modem transceiver, the wireless local area network (WLAN) is dedicated to the coordinate information sharing. In cooperative situations where exchange date rate is not demanding, the wireless serial modem can also be adopted for the coordinate data exchange.

3.1.2 Onboard Tasks Management

As shown in Fig. 7, the onboard software (for each UAV member) consists of the following tasks: (1) sensor information retrieval; (2) cooperative information processing; (3) control algorithms computation; (4) servo driver execution; (5) server and client for cooperative data exchange; (6) wireless communication and (7) data logging. To efficiently manage all of the tasks, we define a simple but reliable executing sequence which is shown in Fig. 8. Specifically:

1. IMU and DAQ are executed first to collect the inflight data from the avionic sensors;
2. SVR is executed to receive the coordinate data;
3. CTL algorithms are calculated;
4. The output control signals are dispatched to the servos (SVO);
5. Data communication and data logging (CLT, CMM, and DLG) are performed.

The execution of all the task threads are managed by the onboard main program. The overall tasks are coordinated to run sequentially. The onboard software is implemented on the QNX Neutrino 6.3.2 real-time operating system (RTOS), which is well suited to the embedded real-time applications. It provides multitasking, threads, priority-driven preemptive scheduling and fast context-switching [19].

From the software design perspective, after identification of the tasks, the corresponding class diagram in UML is shown in Fig. 9. The clsThread is the parent class for

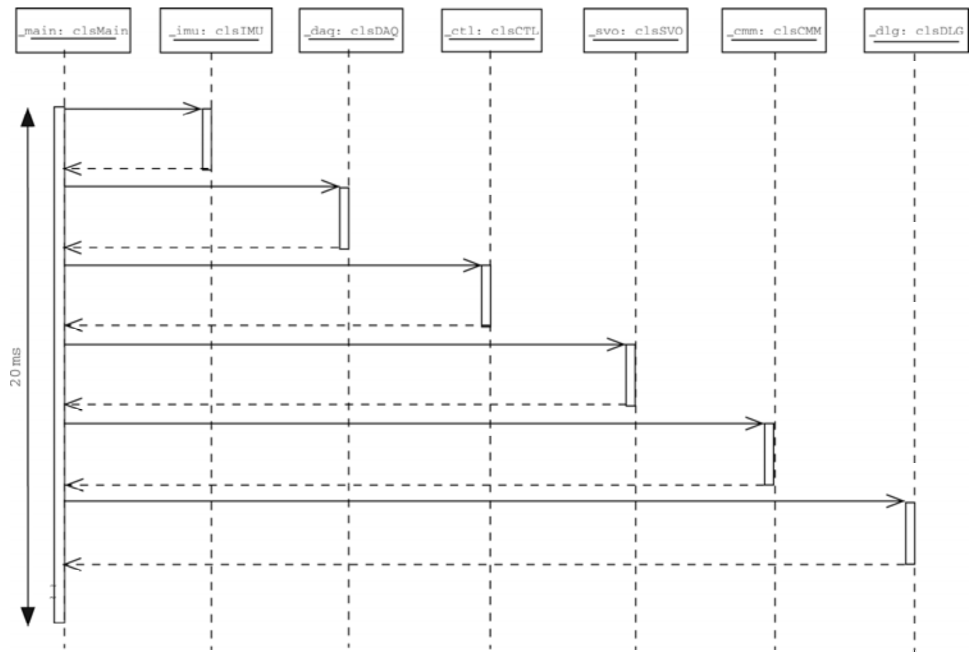


Figure 10. Onboard object sequence diagram.

the six task threads. Details on UML diagrams can be seen in [20].

In addition, the dynamic behaviours of the system objects during the execution period is described. The UML execution sequence diagram is adopted and shown in Fig. 10. It is clear that the main thread manages the working thread in a way that each working thread is activated in a predefined required sequence to accomplish the onboard control loop. The initiated objects during the onboard execution of the each thread is _main, _imu, _daq, _ctl, _svo, _cmm, and _dlg, respectively. All of the above objects is globally active during the whole execution period.

3.1.3 Control Implementation

The control block hires a behaviour-based architecture, which is shown in Fig. 11. The coordinator output $\mathbf{y}_G$ is derived from three input sources. One is the user command from the CMM thread, the second is the coordinate information exchange from the peer to peer network and the last one comes from the environment sensing which is exclusively for precautions in the emergency situation. In our current implementation, the emergency situation is identified based on the status data. In such case, the $\mathbf{y}_G$ instructs the UAV to perform hovering at the current position.

The key components of control block are listed in Table 3. For the inner loop control law, one advanced approach called composite nonlinear feedback (CNF) is adopted. The CNF method has a nice feature that both short settling time and small overshoot can be met. Interested readers on CNF are referred to [21]. In addition, we have implemented the LQR and H_∞ for robust flight control (please see [22] for details). On the other hand, the outer-loop control laws are realized

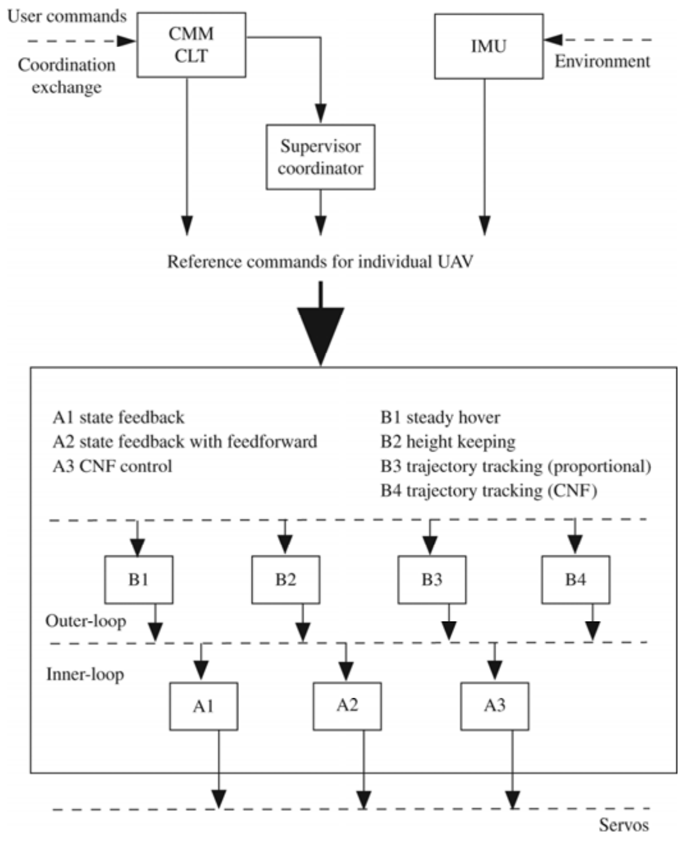


Figure 11. Behaviour-based architecture for control system.

with simple proportional control and CNF control. We note that $\mathbf{u} = (\delta_a, \delta_e, \delta_u, \delta_r)$ contains the control inputs for aileron, elevator, collective, and rudder, respectively. $\mathbf{x} = (u, v, p, q, \phi, \theta, a_s, b_s, w, r, r_{fb})$ is a collection of state variables of the helicopter identified in modelling process and controller design. u, v, and w are the velocity of the

Table 3
Control Components

ID	Function	Description	Parameter
A1	$\mathbf{u} = \mathbf{F}\mathbf{x}$, $\mathbf{x} = \mathbf{x}_{\text{real}} - \mathbf{x}_{\text{ref}}$	State feedback	$\mathbf{x}_{\text{ref}}$
A2	$\mathbf{u} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{v}$	LQR and H_∞	$\mathbf{v}$
A3	$\mathbf{u} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{v} + \rho\mathbf{B}^{\mathbf{T}}\mathbf{P}(\mathbf{x} - \mathbf{H}\mathbf{v})$	CNF	$\mathbf{v}$
B1	$\mathbf{V_g} = \mathbf{k}(\mathbf{X} - \mathbf{X_c})$, $r = k_\psi(\psi - \psi_c)$	Position and direction holding	$\mathbf{X_c}$, ψ_c
B2	$w = k_z(z - z_c)$	Height keeping	z_c
B3	$\mathbf{V_g} = \mathbf{k}(\mathbf{X} - \mathbf{X_c}(\mathbf{t}))$, $r = k_\psi(\psi - \psi_c(t))$	Trajectory tracking	$\mathbf{X_c}(\mathbf{t})$, $\psi_c(t)$
B4	$u = \mathbf{F}\hat{\mathbf{x}}(t) + \mathbf{G}r + \rho\mathbf{F_n}(\hat{\mathbf{x}}(t) - \mathbf{G_e}r)$	Trajectory tracking	$\hat{x}(t)$

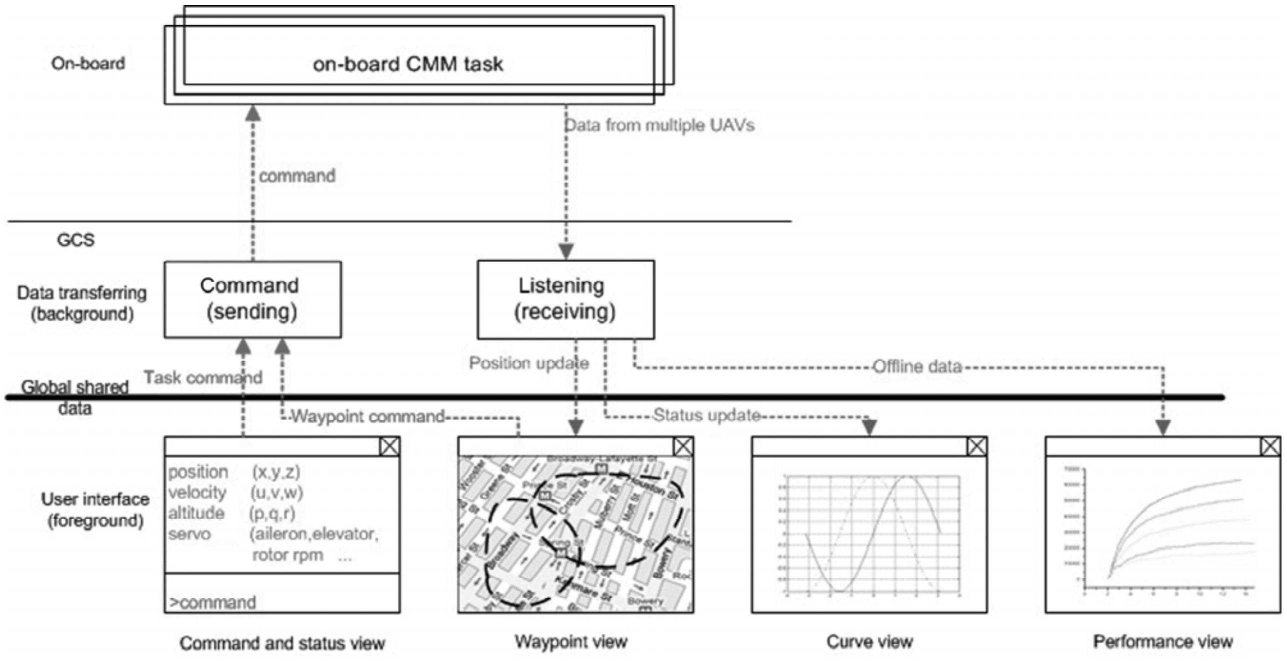


Figure 12. Architecture of ground control station.

helicopter along the three axis in the body frame. Similarly, p, q, and r are the angular rate of roll, pitch and yaw in the body axis. ϕ, θ, and ψ are the roll, pitch and heading angles of the helicopter in the NED frame. Finally, a_s and b_s represent the longitudinal and lateral flapping angles of the main rotor, and r_{fb} is an internal state of the yaw channel. The state variables a_s, b_s, and r_{fb} are estimated by an observer. $\mathbf{v} = (u, v, w, r)_c$ contains a number of parameters representing the target reference (velocity and yaw rate in the body frame) for the inner-loop components. The detailed descriptions of each control component in the inner- and outer-loop can be referred to [23].

3.2 Ground Control Station

The GCS is mainly used for updating new flight tasks or missions to the UAVs, monitoring the status of multiple UAVs and evaluating the performance of coordination control. It is realized with Microsoft Foundation Class (MFC) in a laptop with the Windows XP Professional operating system. The overall architecture is realized *via* the Multiple Document Interface (MDI) approach, which is shown in Fig. 12. The upward and downward arrows indicate the commands transmitted from GCS to the UAVs and the data flow from UAVs to GCS, respectively. We integrate several visual perspectives for the demonstration of flight status data from multiple UAVs. The document class in MFC is for the management of data sending and receiving, and periodic update of all the views consisting of Google Map way point view (we will introduce it later in more detail), status view, command window, gauge view and curve view.

The UML class diagram of the main view classes is listed in Fig. 13. All the views accept the global data received from the receiving thread and displayed with an update frequency of 1 Hz.

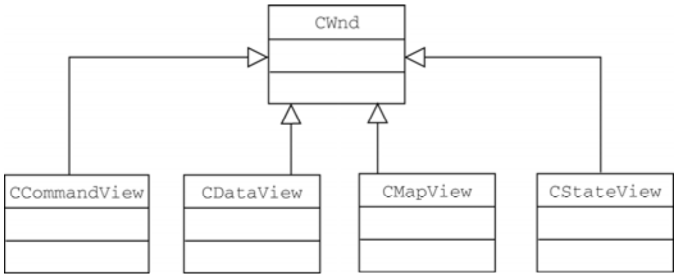
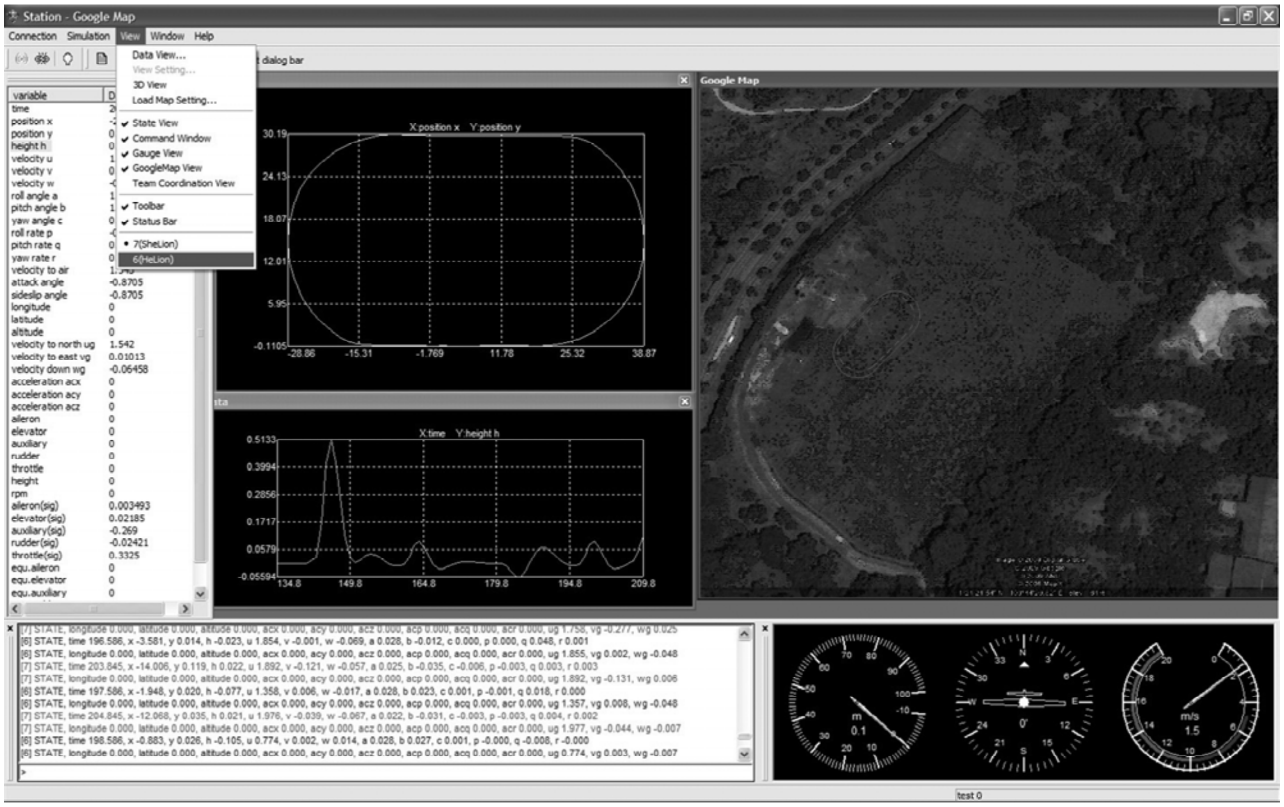


Figure 13. UML class diagram of the ground control station.



variable	D
time	2
position x	-
position y	0
height h	0
velocity u	1
velocity v	0
velocity w	-
roll angle a	1
pitch angle b	1
yaw angle c	0
roll rate p	-
pitch rate q	0
yaw rate r	0
velocity to air	1.5
attack angle	-0.8705
sideslip angle	-0.8705
longitude	0
latitude	0
altitude	0
velocity to north ug	1.542
velocity to east vg	0.01013
velocity down wg	-0.06458
acceleration acx	0
acceleration acy	0
acceleration acz	0
aileron	0
elevator	0
auxiliary	0
rudder	0
throttle	0
height	0
rpm	0
aileron(sig)	0.003493
elevator(sig)	0.02185
auxiliary(sig)	-0.269
rudder(sig)	-0.02421
throttle(sig)	0.3325
equ.aileron	0
equ.elevator	0
equ.auxiliary	0

```
[7] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 1.758, vg -0.277, wg 0.025
[6] STATE, time 196.586, x -3.581, y 0.014, h -0.023, u 1.854, v -0.001, w -0.069, a 0.028, b -0.012, c 0.000, p 0.000, q 0.048, r 0.001
[6] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 1.855, vg 0.002, wg -0.048
[7] STATE, time 203.845, x -14.006, y 0.119, h 0.022, u 1.892, v -0.121, w -0.057, a 0.025, b -0.035, c -0.006, p -0.003, q 0.003, r 0.003
[7] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 1.892, vg -0.131, wg 0.006
[6] STATE, time 197.586, x -1.948, y 0.020, h -0.077, u 1.358, v 0.006, w -0.017, a 0.028, b 0.023, c 0.001, p -0.001, q 0.018, r 0.000
[6] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 1.357, vg 0.008, wg -0.048
[7] STATE, time 204.845, x -12.068, y 0.035, h 0.021, u 1.976, v -0.039, w -0.067, a 0.022, b -0.031, c -0.003, p -0.003, q 0.004, r 0.002
[7] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 1.977, vg -0.044, wg -0.007
[6] STATE, time 198.586, x -0.883, y 0.026, h -0.105, u 0.774, v 0.002, w 0.014, a 0.028, b 0.027, c 0.001, p -0.000, q -0.008, r -0.000
[6] STATE, longitude 0.000, latitude 0.000, altitude 0.000, acx 0.000, acy 0.000, acz 0.000, acp 0.000, acq 0.000, acr 0.000, ug 0.774, vg 0.003, wg -0.007
>
```

Figure 14. Graphical interface of ground control station.

The GCS consists of two layers: (1) the background layer which involves data receiving and sending with the onboard computer and (2) foreground layer which contains various in-flight data views and a command window. Specifically:

1. The background layer has mainly two tasks, receiving flight status from and sending commands to multiple UAVs, both of which interact with the UAV onboard CMM task. The receiving thread accepts all the data from the fleet of UAVs and identify each status data *via* the telegraph packet header;
2. Consequently, the corresponding multiple display is executed in the foreground layer. The cooperative way points of the paths are demonstrated. Similarly, the upload link can broadcast the commands to all UAVs, or alternatively send commands to a specific UAV, both *via* the sending task. The global status data from UAVs are dynamically updated from the background layer. To make the GCS more user-friendly, we incorporate the Google Map view to better demonstrate the cooperative behaviours of the fleet of multiple UAVs. We have captured several maps of the flying fields where we plan to conduct the flight tests from Google Earth and record necessary GPS data on the corners of these maps. In the flight test, the GPS signal from the onboard system will keep updated on the global shared data, and the cooperative paths of multiple UAVs are displayed on the Google Map way point view. For indoor flight test, since the GPS signal is not available, we can manually set the position information to simulate this functionality in the way point view. Fig. 14 is a snapshot of our ground control station. If multiple UAVs are connected with the GCS, we can freely select any UAV helicopter (currently HeLion or SheLion) to be displayed on all of the views.

3.3 Formation Flight Implementation

Based on the onboard software and GCS, the initial leader–follower formation flight is implemented in a centralized

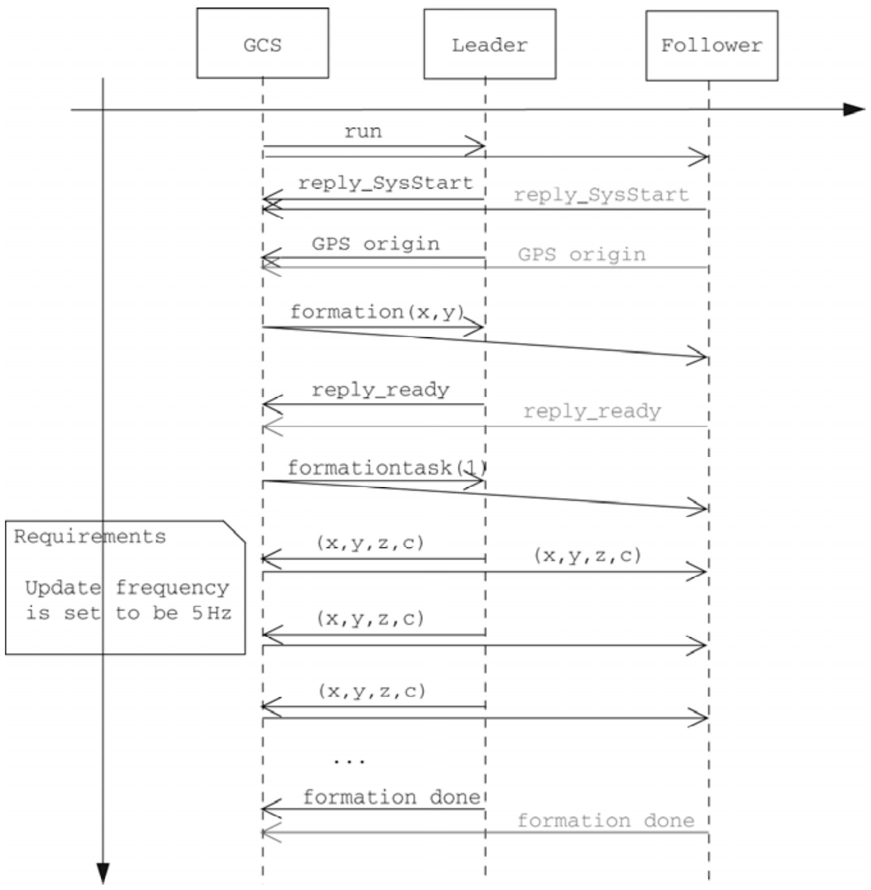


Figure 15. Message sequence diagram in formation flight.

approach, in which the GCS is configured to be the central node performing the supervisory and coordinator role. Specifically, the CMM thread on leader sends its status data back to GCS with a rate of ten times per second. Once the GCS receives the update of the leader, the supervisor derives the next state (task) for each UAV and sends it to coordinator for task dispatch and transmits the output to the follower. Then the follower is assigned a task of performing proper behaviour based on the current status and reference signal. The overall procedure is illustrated in Fig. 15.

In each flight experiment, the initial GPS locations of both leader and follower are required to send to GCS. Based on these information, GCS then derives the reference position in the coordinate of the follower and updates the transformed position to the follower(s). Next, GCS issues a command to specify the rendezvous position where the formation flight will start. The leader and follower then perform "headto" command to hover at the specified positions to start formation. Once both the leader and follower hover at the reference rendezvous position, they send the "ready" information to GCS. Following this step, GCS broadcasts the "formationtask" command to both UAVs and the two UAVs start performing trajectory tracking task to complete the formation flight. In the whole formation flight procedure, GCS is responsible for monitoring the threshold condition of collision avoidance periodically.

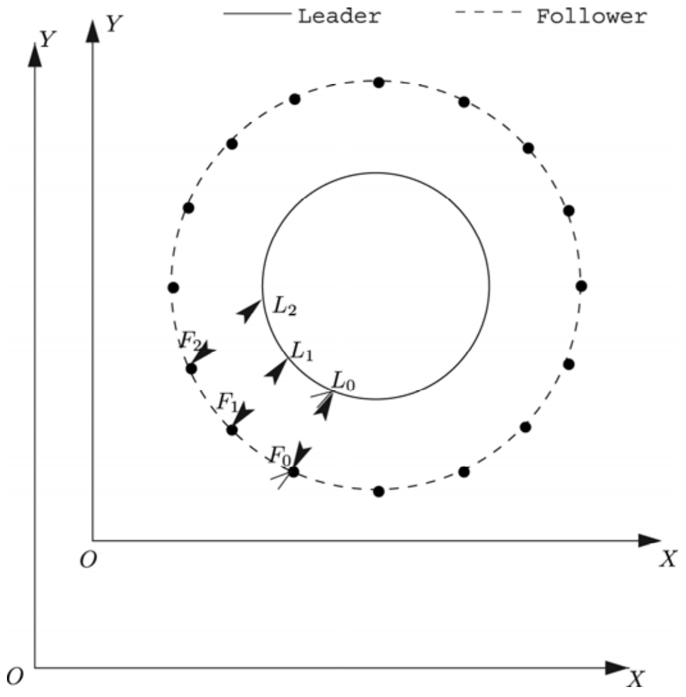
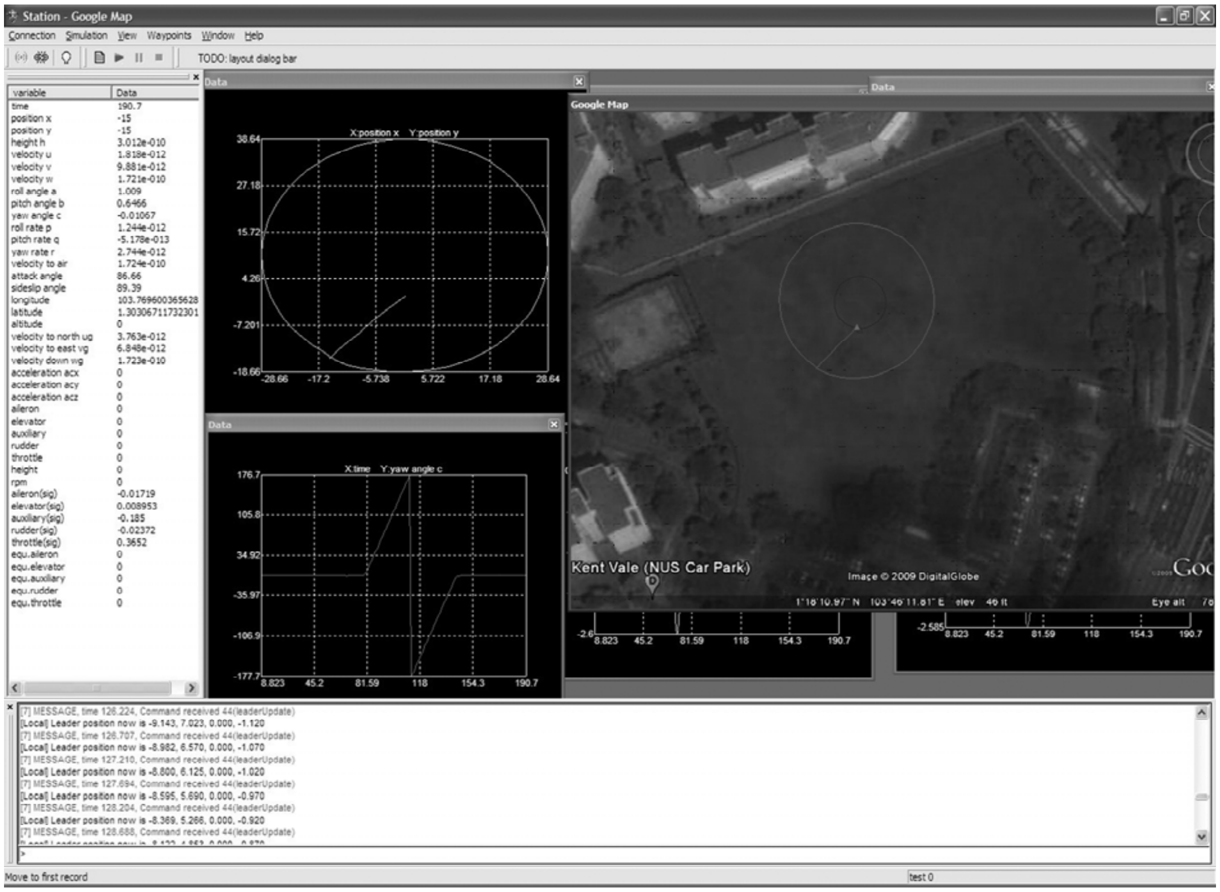


Figure 16. Leader–follower circle formation scenario.

Considering the $\pm 3\,\mathrm{m}$ CEP accuracy of currently adopted GPS receiver and necessary redundancy, the boundary set during formation flight is $\pm 10\,\mathrm{m}$ to guarantee the overall safety.


Station - Google Map
Connection Simulation View Waypoints Window Help
TODO: layout dialog bar
Data
X:position x Y:position y
X:time Y:yaw angle c
Google Map
Kent Vale (NUS Car Park)
Image © 2009 DigitalGlobe
1°18'10.97" N 103°46'11.81" E elev 46 ft
Eye alt
test 0

variable	Data
time	190.7
position x	-15
position y	-15
height h	3.012e-010
velocity u	1.818e-012
velocity v	9.881e-012
velocity w	1.721e-010
roll angle a	1.009
pitch angle b	0.6466
yaw angle c	-0.01067
roll rate p	1.244e-012
pitch rate q	-5.178e-013
yaw rate r	2.744e-012
velocity to air	1.724e-010
attack angle	86.66
sideslip angle	89.39
longitude	103.769600365628
latitude	1.30306711732301
altitude	0
velocity to north ug	3.763e-012
velocity to east vg	6.848e-012
velocity down wg	1.723e-010
acceleration acx	0
acceleration acy	0
acceleration acz	0
aileron	0
elevator	0
auxiliary	0
rudder	0
throttle	0
height	0
rpm	0
aileron(sig)	-0.01719
elevator(sig)	0.008953
auxiliary(sig)	-0.185
rudder(sig)	-0.02372
throttle(sig)	0.3652
equ.aileron	0
equ.elevator	0
equ.auxiliary	0
equ.rudder	0
equ.throttle	0

```
[7] MESSAGE, time 126.224, Command received 44(leaderUpdate)
[Local] Leader position now is -9.143, 7.023, 0.000, -1.120
[7] MESSAGE, time 126.707, Command received 44(leaderUpdate)
[Local] Leader position now is -8.982, 6.570, 0.000, -1.070
[7] MESSAGE, time 127.210, Command received 44(leaderUpdate)
[Local] Leader position now is -8.800, 6.125, 0.000, -1.020
[7] MESSAGE, time 127.694, Command received 44(leaderUpdate)
[Local] Leader position now is -8.595, 5.690, 0.000, -0.970
[7] MESSAGE, time 128.204, Command received 44(leaderUpdate)
[Local] Leader position now is -8.369, 5.266, 0.000, -0.920
[7] MESSAGE, time 128.688, Command received 44(leaderUpdate)
```

Move to first record

Figure 17. Leader–follower formation in the GCS.

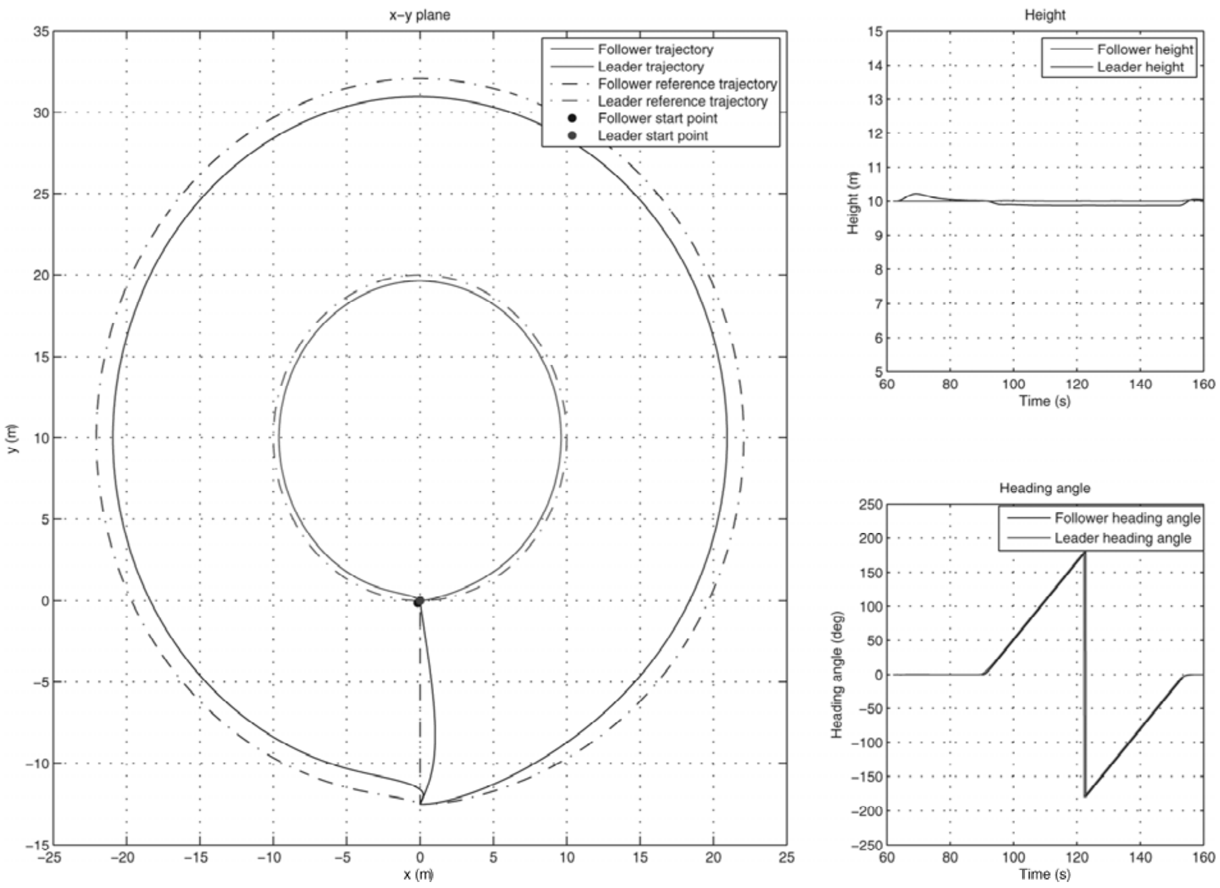


Figure 18. Leader–follower in a circle formation.

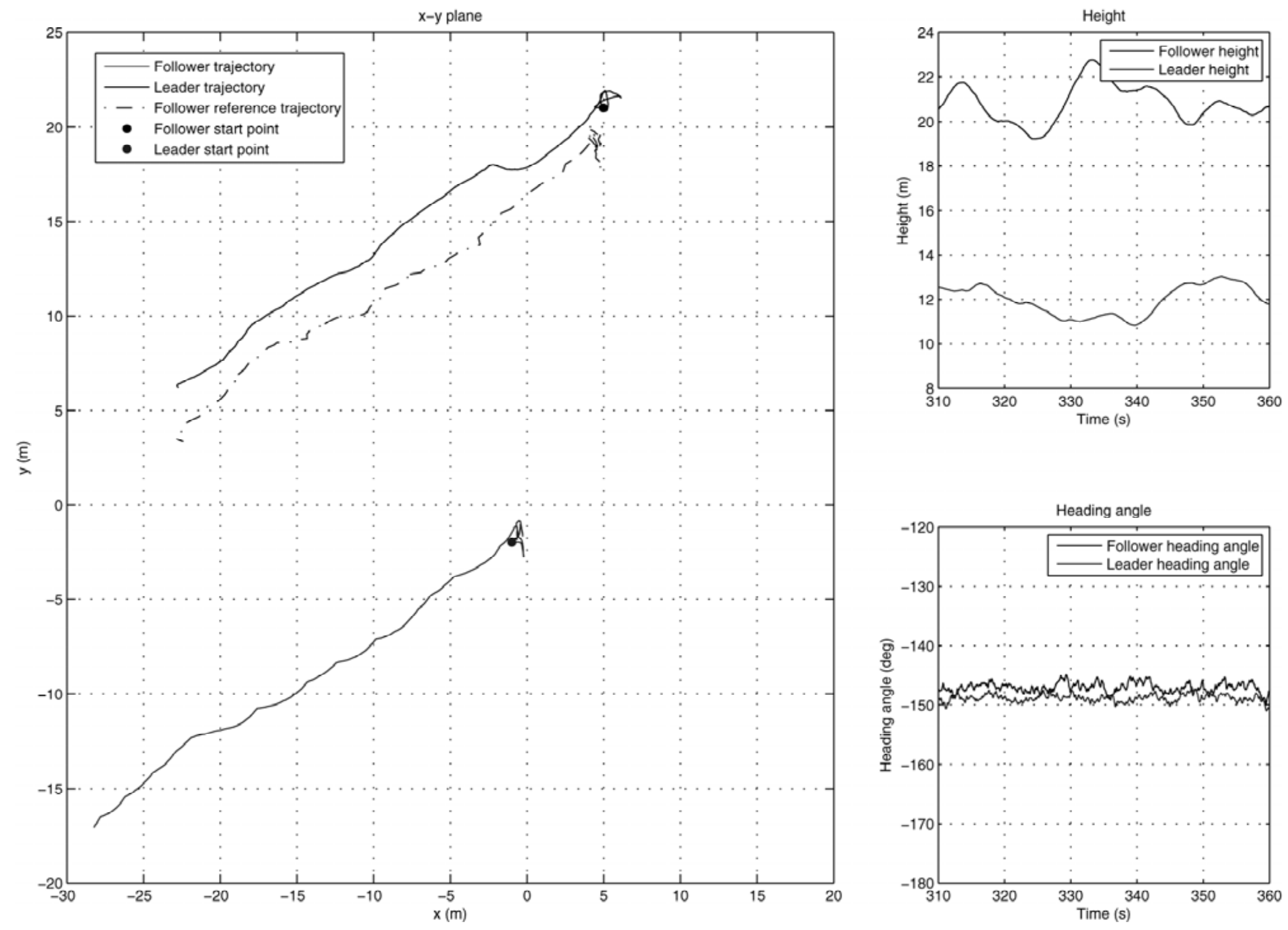


Figure 19. Leader–follower in a line formation.

4. Simulation and Practical Implementation

In this section, we present the simulation and practical implementation results of formation flight based on the proposed framework and developed comprehensive software system.

4.1 Simulation

Preflight simulation is important for evaluating the overall behaviours of both onboard software and GCS. With a built-in UAV model, we carry out the hardware-in-the-loop simulation (HILS) in which the interactions among multiple UAVs and GCS and precautions under different failure situations are tested. Note that the velocities and positions are generated by a simulation block which is based on a verified aerodynamic model for our UAV helicopters.

For the case of formation flight, the leader is commanded to perform a predefined path tracking, and the follower is required to follow the leader with a constant distance offset along both longitudinal and lateral directions in the coordinate of the leader. HeLion is assigned as the leader and SheLion performs as the follower. Various scenarios have been evaluated and here we adopt a circle path tracking as the example, which is shown in Fig. 16. Note that L_0 and F_0 are the initial reference rendezvous positions for the leader and the follower, respectively. The points L_i $(i = 1, 2, \ldots, N)$ refer to the predefined trajectory for the leader, and the points F_i $(i = 1, 2, \ldots, N)$ refer to the reference points that the follower receives from the coordinator of the GCS.

The simulation results are depicted in Fig. 17 and 18. In the simulation, the initial starting point of the leader and follower can be determined on the Google Map. The information exchange frequency is set as 5 Hz and the leader is commanded to perform a circle path tracking with a radius of 10 m with an tangential velocity of 1 m/s. It is clear to see that in Fig. 18, the follower tracks the reference path given from the GCS accurately. The tracking error between the trajectory of the follower and its reference is due to the minor inherent delay in tracking control. We also note that since the distance between the leader and the follower is less than the required distance at first, the follower performs a rendezvous task such that two UAVs are ready for the formation task. Fig. 18 also indicates that the heading angle tracking is also sufficiently accurate. Such simulation results provide us with enough confidence to conduct the formation flight practically.

4.2 Practical Implementation

In practical implementation, we first conduct the line-path based formation flight considering the safety of the helicopters. In this formation flight scenario, the leader performs a line path with the length of 30 m and 1 m/s constant forward speed. The lateral distance is set as

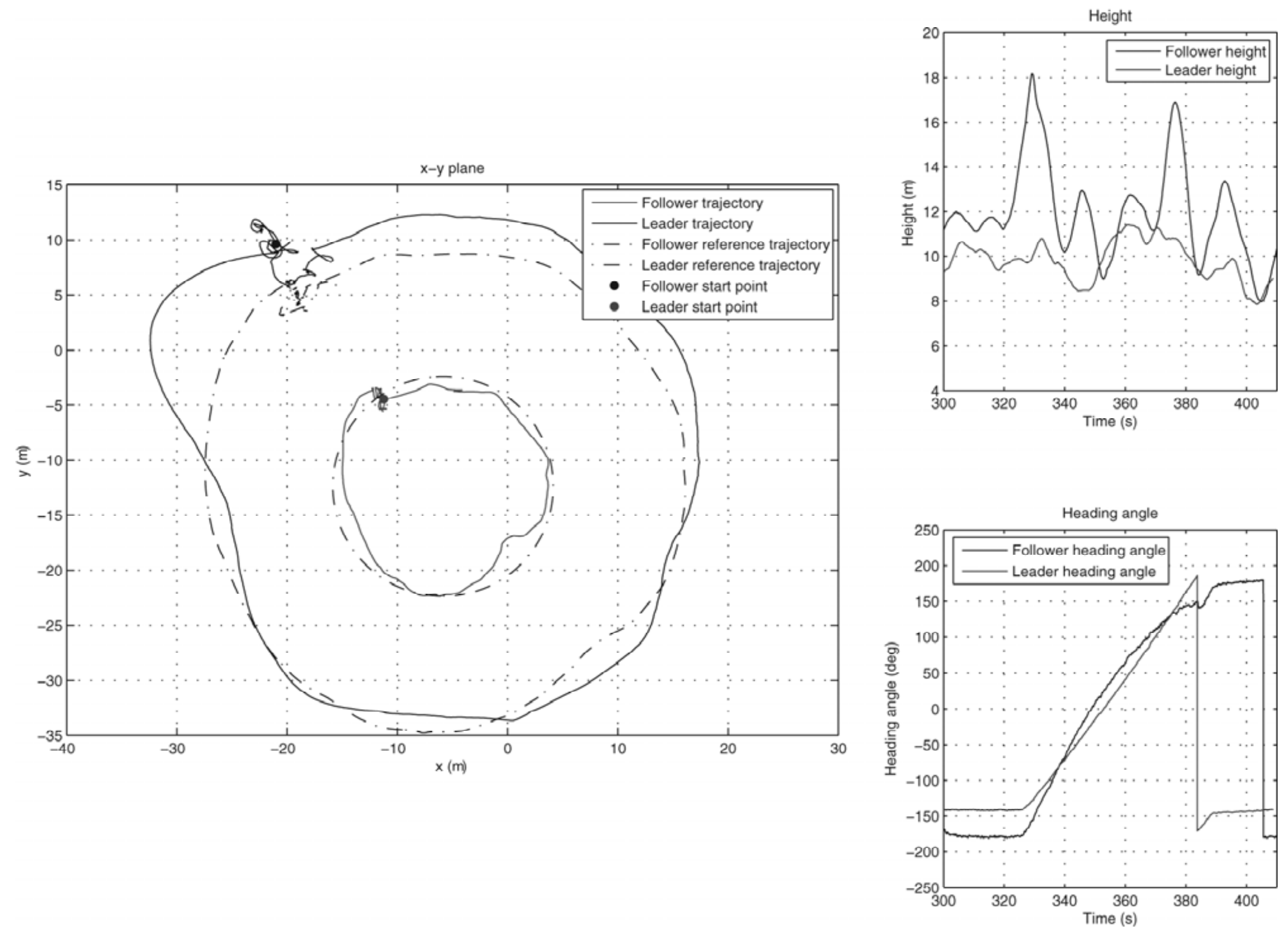


Figure 20. Leader–follower in a circle path based formation flight.

Figure 21. Leader and follower in formation flight.

15 m to ensure the overall safety. Both the leader and the follower perform a height-keeping flight. The flight test results are shown in Fig. 19. We note that the basic line path formation flight is successfully achieved with our developed system. The small fluctuations in the heading angle is due to the minor measurement inaccuracy of the inertial navigation sensors.

After the basic line path based formation is accomplished, we performed a circle path based formation, and the flight tests are listed in the Fig. 20.

1. The concentric-circles formation flight is well completed. For the follower SheLion, it can closely track the reference trajectory in the whole formation flight procedure. In the ending part, the tracking performance is degraded, which is mainly caused by the strong wind gust;
2. Regarding the height, both HeLion and SheLion are required to maintain a constant value of 10 m. From Fig. 20, it can be noted that the HeLion's height is well maintained, with the accuracy of ±1 m. The height control of SheLion is worse than that for HeLion, which is mainly caused by the following two reasons:
   (1) The reference signal of SheLion is the measurement generated by HeLion's navigation sensor and with measurement error involved; at time points 329 s and 377 s, She Lion has experienced a height fluctuation caused by the loose lock of GPS signal.
   (2) The small gap in the heading angle measurements is caused by the improper calibration of the navigation sensors. It is unavoidable for the low-cost navigation sensors we currently adopt. A correction function has been used to reduce such a hardware bias.

Finally, a nice snapshot of the leader and the follower during the formation flight is captured in Fig. 21.

5. Conclusions

In this paper, an efficient framework of coordination control for multiple UAVs has been presented and explained. Based on this framework, we have developed a comprehensive software for the practical coordination control implementations. For the two key components of the software, the onboard software architecture can fulfill the need of realizing real-time cooperative behaviour among multiple UAVs and the ground control station is also qualified for monitoring and commanding multiple UAVs. Both the simulation and practical implementation for formation flight have been successfully conducted to prove the efficiency of the proposed framework and developed software. In the next stage, we tend to realize the full-envelope formation flight including automatic takeoff and landing and other more complicated cooperative flight missions.

References

[1] C.J. John, Automatic formation flight control system (AFFCS) – A system for automatic formation flight control of vehicles not limited to aircraft, helicopters, or space platforms, *U.S. Patent 6,926,233*, September 8, 2005.

[2] D. Fox, W. Burgard, H. Kruppa, & S. Thrun, A probabilistic approach to collaborative multi-robot localization, *Autonomous Robots*, *8*(3), 2000, 325–344.

[3] W. Burgard, M. Moors, D. Fox, R. Simmons, & S. Thrum, Collaborative multi-robot exploration, *Proceedings IEEE International Conference Robotics and Automation*, 2000, 476–481.

[4] G. Dedeoglu & G.S. Sukhatme, Landmark-based matching algorithm for cooperative mapping by autonomous robots, *Proceedings of the 5th International Symposium Distributed Autonomous Robotics Systems*, 2000.

[5] J. Feddema & D. Schoenwald, Decentralized control of cooperative robotic vehicles, Presented at the SPIE, *4364*, Aerosense, Orlando, FL, 2001.

[6] J.S. Jennings, G. Whelan, & W.F. Evans, Cooperative search and rescue with a team of mobile robots, *Proceedings of IEEE International Conference Advanced Robotics*, 1997, 193–200.

[7] D. Rus, B. Donald, & J. Jennings, Moving furniture with teams of autonomous robots, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, 235–242.

[8] D. Stilwell & J. Bay, Toward the development of a material transport system using swarms of ant-like robots, *Proceedings of IEEE International Conference Robotics and Automation*, Atlanta, GA, 1993, 766–771.

[9] T. Shima & S. Rasmussen, *UAV cooperative decision and control, challenges and practical approaches*, (SIAM: Philadelphia, 2009).

[10] J.P. How, Multi-vehicle flight experiments: Recent results and future directions, *Proceedings of AVT-146 Symposium on Platform Innovations and System Integration for Unmanned Air, Land and Sea Vehicles*, Florence, Italy, 2007.

[11] J.P. How, E. King, & Y. Kuwata, Flight demonstrations of cooperative control for UAV teams, *Proceedings of the 3rd AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, Chicago, IL, AIAA-2004-6490, 2004.

[12] G. Hoffmann, D.G. Rajnarayan, S.L. Waslander, *et al.*, The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC), *23rd Digital Avionics Systems Conference*, *2*, 2004, 12.E.4.1–12.E.4.10.

[13] S. Bayraktar, G.E. Fainekos, & G.J. Pappas, Hybrid modeling and experimental cooperative control of multiple unmanned aerial vehicles, *43rd IEEE Conference on Decision and Control*, Atlantis, Bahamas, *4*, 2004, 4292–4298.

[14] T.W. McLain & R.W. Beard, Unmanned air vehicle testbed for cooperative control experiments, *Proceedings of American Control Conference*, Boston, MA, *6*, 2004, 5327–5331.

[15] A. Attoui, *Real-Time and Multi-Agent Systems*, (Springer-Verlag: Berlin, 2000).

[16] M. Dong, B.M. Chen, G. Cai, & K. Peng, Development of a real-time onboard and ground station software system for a UAV helicopter, *Journal of Aerospace Computing, Information and Communication*, *4*, 2007, 933–955.

[17] R.W. Beard, J. Lawton, & F.Y. Hadaegh, A coordination architecture for spacecraft formation control, *IEEE Transactions on Control Systems Technology*, *9*(6), 2001, 777–790.

[18] G. Cai, B.M. Chen, K. Peng, M. Dong, & T.H. Lee, Modeling and control system design for a UAV helicopter, *Proceedings of the 14th Mediterranean Conference on Control Automation*, Ancona, Italy, 2006, 1–6.

[19] QNX Neutrino RTOS v6.3, System Architecture, Sixth Edition, QNX Software Systems Corporation.

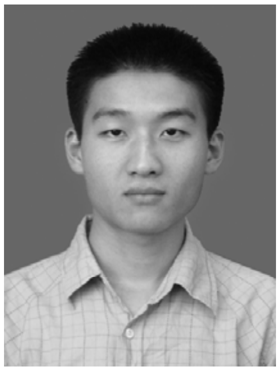
[20] B.P. Douglass, *Real-time UML: Developing efficient objects for embedded systems*, (Addison-Wesley, MA, 1999).

[21] B.M. Chen, T.H. Lee, K. Peng, & V. Venkataramanan, Composite nonlinear feedback control for linear systems with input saturation: Theory and an application, *IEEE Transaction on Automatic Control*, *48*(3), 2003, 427–439.

[22] G. Cai, B.M. Chen, & T.H. Lee, Design and implementation of robust flight control system for a small-scale UAV helicopter, *7th Asian Control Conference*, Hong Kong, 2009, 691–697.

[23] A. Ryan, X. Xiao, & S. Rathinam, *et al.*, A modular software infrastructure for distributed control of collaborating UAVs, *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, Colo, *5*, 2006, 3248–3256.

Biographies



Mr. Xiangxu Dong received his B.S. degree from the Department of Communications Engineering at Xiamen University in 2006. Since 2007, he has been a Ph.D. candidate at National University of Singapore. His research interests include real-time software, cooperative coordination, formation control and unmanned aerial vehicles.

Ben M. Chen received his B.S. degree in mathematics and computer science from Xiamen University, China, in 1983, M.S. degree in electrical engineering from Gonzaga University, USA, in 1988, and Ph.D. degree in electrical and computer engineering from Washington State University, USA, in 1991. He was an assistant professor from 1992 to 1993 at the State University of New York at Stony Brook, USA. Since 1993, he has been with the Department of Electrical and Computer Engineering, National University of Singapore, where he is currently a full professor. His current research interests are in systems theory, robust control, unmanned aerial systems, and financial market modeling. Dr. Chen is an IEEE Fellow. He is the author/co-author of 8 research monographs including Robust and H Control (Springer, New York, 2000), Hard Disk Drive Servo Systems (Springer, New York, 1st Ed., 2002; 2nd Ed., 2006), Linear Systems Theory (Birkhäuser, Boston, 2004), and Unmanned Rotorcraft Systems (Springer, New York, in press). He has served on the editorial boards of a number of journals including IEEE Transactions on Automatic Control, Systems & Control Letters, Automatica, and Journal of Control Theory and Applications. He was the recipient of the Best Poster Paper Award, 2nd Asian Control Conference, Seoul, Korea (1997); University Researcher Award, National University of Singapore (2000); IES Prestigious Engineering Achievement Award, Institution of Engineers, Singapore (2001); Temasek Young Investigator Award, Defence Science & Technology Agency, Singapore (2003); Best Industrial Control Application Prize, 5th Asian Control Conference, Melbourne, Australia (2004); Best Application Paper Award, 7th Asian Control Conference, Hong Kong, China (2009); and Best Application Paper Award, 8th World Congress on Intelligent Control and Automation, Jinan, China (2010).

Dr. Guowei Cai has received his B.E. degree in electrical and electronics engineering from Tianjin University, Tianjin, China, in 2002 and the Ph.D. degree in electrical and computer engineering from National University of Singapore, Singapore, in 2009. From 2008 to 2009, he was a Research Fellow in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Since 2009, he has been a Research Scientist in Temasek Laboratories, National University of Singapore. His current research interests include construction, modelling identification, control theory application, and formation control of small-scale fixed-wing and rotorcraft UAV systems. He was a recipient of the Best Application Paper Prize at the 7th Asian Control Conference, Hong Kong, China (2009).

Dr. Hai Lin is currently an assistant professor in the National University of Singapore, Electrical and Computer Engineering Department. He received his B.S. degree from University of Science and Technology, Beijing, China in 1997, the M.Eng. degree from Chinese Academy of Science, China in 2000, and the Ph.D. degree from the University of Notre Dame, USA in 2005. He is the chair of the IEEE SMC Singapore Chapter since 2009, and serves in several editorial board and conference organizing committee. His research interests are in the multidisciplinary study of the problems at the intersection of control, communication, computation and life sciences. His current research thrust is on hybrid control systems, multi-robot coordination and systems biology.

Tong H. Lee received his B.A. degree with First Class Honours in the Engineering Tripos from Cambridge University, England, in 1980; and the Ph.D. degree from Yale University in 1987. He is a Professor in the Department of Electrical and Computer Engineering at the National University of Singapore (NUS); and also a Professor in the NUS Graduate School, NUS NGS. He was a Past Vice-President (Research) of NUS. Dr. Lee's research interests are in the areas of adaptive systems, knowledge-based control, intelligent mechatronics and computational intelligence. He currently holds Associate Editor appointments in the *IEEE Transactions in Systems, Man and Cybernetics; IEEE Transactions in Industrial Electronics; Control Engineering Practice* (an IFAC journal) and the *International Journal of Systems Science* (Taylor and Francis, London). In addition, he is the Deputy Editor-in-Chief of IFAC Mechatronics journal. Dr. Lee was a recipient of the Cambridge University Charles Baker Prize in Engineering; the 2004 ASCC (Melbourne) Best Industrial Control Application Paper Prize; the 2009 IEEE ICMA Best Paper in Automation Prize; and the 2009 ASCC Best Application Paper Prize. He has also co-authored five research monographs (books), and holds four patents (two of which are in the technology area of adaptive systems, and the other two are in the area of intelligent mechatronics). He has published more than 300 international journal papers. Dr. Lee was an Invited Panelist at the World Automation Congress, WAC2000 Maui U.S.A.; an Invited Keynote Speaker for IEEE International Symposium on Intelligent Control, IEEE ISIC 2003 Houston U.S.A.; an Invited Keynote Speaker for LSMS 2007, Shanghai China; an Invited Expert Panelist for IEEE AIM2009; an Invited Plenary Speaker for IASTED RTA 2009, Beijing China; an Invited Keynote Speaker for LSMS 2010, Shanghai China and an Invited Keynote Speaker for IASTED CA 2010, Banff, Canada.