2009 IEEE International Conference on Control and Automation
Christchurch, New Zealand, December 9-11, 2009

Development of a Comprehensive Software System for Implementing Cooperative Control of Multiple Unmanned Aerial Vehicles

Xiangxu Dong, Ben M. Chen, Guowei Cai, Hai Lin, Tong H. Lee
Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576
Email: {dong07, bmchen, elelh, eleleeth}@nus.edu.sg

Abstract— **In this work, we focus on establishing a framework and developing a comprehensive real-time software platform for verifying and realizing flight coordination among multiple unmanned aerial vehicles (UAVs). The framework is capable of providing flexible architecture for design of cooperative control laws. The overall software platform incorporates the onboard real-time software for UAVs and that for the ground control station. It employs a distributed architecture to facilitate the deployment of experiments with multiple unmanned vehicles, efficient monitoring and commanding the UAVs from the ground station. The system has been successfully tested in the hardware-in-the-loop simulation and in actual flight formation experiment involving multiple UAVs.**

I. Introduction

In recent years, more research efforts have been focused on the development of cooperative behaviors among multiple unmanned aerial vehicles in both military and civilian applications. The multiple vehicles possess more powerful capability when executing certain tasks in a cooperative way than in the single UAV. The potential application scenarios may include urban collaborative surveillance, geographic mapping, mobile sensor network, emergent rescue and fire detection, etc. However, all these scenarios propose critical requirements for the real time software system design. Due to the real-time nature of these tasks, all tasks must be finished within a predetermined specifiable time boundary with an acceptable quality of service (QoS), and possibly in a distributed approach in a multi-agent systems (MAS). Hence, the software design plays a critical role in the overall system performance. The architectures of communication, control and sensing are the three main principle aspects of the cooperative multi-agent systems [6].

Many UAV research groups have developed their own MAS system platforms. In [2], the authors develop the software architecture based on the Server-Client with inter-process communication and synchronization mechanisms. Also in [3], the authors propose to use a Datahub to resolve the issue of data consistency and inter-process messaging. Similarly, the use of Datahub can also be found in [4]. In the MIT group, the platform is more complex as their hardware-in-the-loop configuration consists of up to eight aircrafts and a cluster of planning CPUs as well. There are also other valuable platforms worldwide focusing on the coordinate control of a fleet of UAVs. However, the design of achieving modularity and efficiency of the overall software system is a nontrivial task.

This paper first describes the overall system architecture for coordination and control in Section II. Section III gives a detailed description of software architecture for both distributed onboard systems and the ground control station (GCS). Section IV presents the formation flight results of two UAVs both in indoor simulation and outdoor tests, with conclusions given in section V.

II. Systems Overall Architecture

A. Introduction

The overall scenario of cooperative control and coordination of multiple UAVs is shown in Fig. 1. Our current aim is to develop a distributed real-time multi-agent systems which can accomplish a task in a real-time, distributed and cooperative approach. All UAVs are primarily identical both in the configuration of hardware and software. And in coordinated scenarios, such as in cooperative formation control application, one can be assigned as a leader while the other two are assigned as two followers. The ground station has the capabilities of monitoring and controlling all the UAVs simultaneously. All the flight information of individual UAV and cooperated behaviors status (such as flight path, communication activities) are available for the ground operator. Besides, the ground operator can send commands to control one individual UAV or broadcast commands to the fleet to execute a cooperative fleet behavior. With such systems, the user specifies a task and all the entities of this infrastructure will finish the assigned task in an intelligent, distributed and cooperative manner.

B. Architecture of Coordination and Control

The architecture of the coordination and control for multiple UAVs determines the overall performance of the system, such as efficiency, stability, scalability, modularity and etc. Thus, the coordinate architecture should be organized in hierarchical layers to accommodate requirements as much as possible. There are three abstract layers. The highest layer coordinates the dynamic transitions from one state to another in the overall coordination task. For example, in the formation flight, all the UAVs will first finish the rendezvous action before being ready

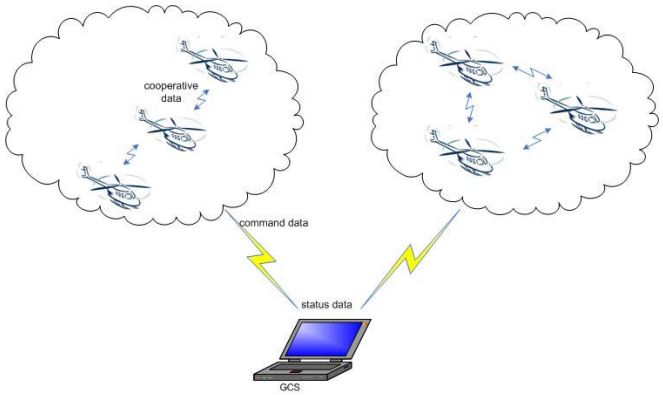

Fig. 1. Scenario of multi-UAV systems.

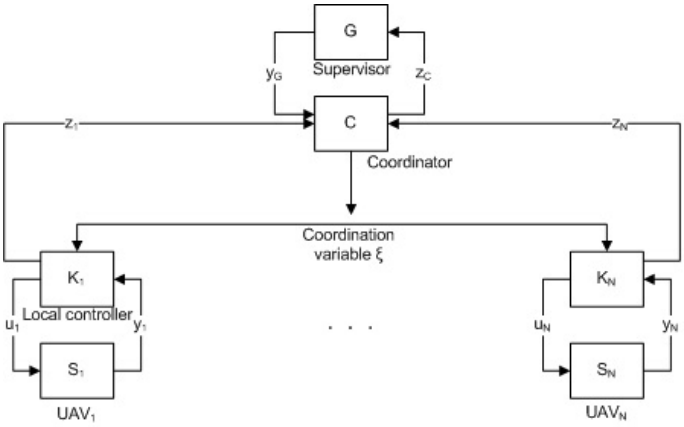

Fig. 2. Architecture for coordinated control among multiple UAVs.

for the following formation flight task. The next abstract layer is for coordination task dispatch based on the coordination mechanism: to assign the proper task to the corresponding UAV. The bottom level is to realize full automatic control based on its assigned task. The idea of the adopted architecture for our application-oriented project comes from [1]. Fig. 2 illustrates the block diagrams of the proposed architecture in our multiple-UAV system. In this architecture, $\mathcal{S}_i$ represents the dynamic model of the ith UAV, with the control input vector $\mathbf{u}_i$ and the measurable output vector $\mathbf{y}_i$.

The higher layer of the local UAV is the coordinator $\mathcal{C}$. It receives coordinate performance input vector from all or selected UAVs, then process and encapsulate the performance evaluation result vector z_C to the top layer according to the coordination mechanism. And it will adjust its 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 broadcast or multicast to the UAV team. In the case of leader-follower based formation flight, the information of the leader is the coordination mechanism. In other words, the coordinator will dispatch the tasks to each follower based on the leader information update. The system $\mathcal{G}$ locating at the highest level is a discrete-event system, which acts as a supervisor to regulate the performance of coordination for the multiple-UAV system.

C. Information Architecture

The information architecture in the coordination system plays the role of broadcasting or multicasting the coordination information to the UAVs. The information flow includes communication data, control data and coordination data.

1) Centralized Architecture: With centralized network architecture, all the information of the UAVs are sent back to a central node and processed at this node (such as GCS). This approach can reach global optimization with the cost of burdening the central node both in communication and processing, while there is no information exchange among the UAVs. All the information exchange is buffered and transmitted via the central node, GCS. This information exchange architecture is applicable to the situation where coordination information flow is less or the central node has powerful processing unit. In addition, the centralized form facilitate the information consistency and ease for tracking and analyzing the performance of coordination and control.

2) Decentralized Architecture: With decentralized architecture, each UAV node shares the information of its neighborhood nodes. Thus the global coordinate behaviors can be sub-optimal instead of global optimal if the number of UAVs is large. On the other side, with this distributed architecture, the overall coordinate system is more robust and scalable than that with the centralized architecture in case of node lost and information topology change. In this architecture, the information exchange only happens among peer nodes where coordinate behaviors are accomplished. In this architecture, the GCS plays the role of only monitoring the flight status of all the UAVs without participation in the coordination of the cooperative behavior.

3) Hybrid Architecture: With hybrid architecture, the overall multi-agent systems consist of both decentralized and centralized architectures to achieve a trade off in the global performance as well as local burden in communication and computing. In the initial development of coordinate behaviors, the GCS acts as the central node to assist the fulfillment of cooperative behavior while the coordination information exchange is realized in the decentralized approach. Also, with the assistance from the GCS in the coordination procedure, the coordination status of the team can be reflected in real-time in the GCS for monitoring and analysis.

III. Software Development

A. UAV Onboard System

The current fully developed helicopters in our research team at the National University of Singapore are named HeLion and SheLion, respectively. Both of them have realized full envelop automatic flight, including automatic vertical take off and landing, path tracking and vision-based tracking. The onboard system is composed of a PC/104 computer stack, an

inertial measurement unit (IMU), a data acquisition board, a sonar chip, a wi-fi card, a wireless communication module and servo drivers [5]. Our developed helicopter is shown in Fig. 3.

Fig. 3. Raptor 90 helicopter.

1) Onboard System Tasks and Architecture: For real-time systems, the one-process-multiple-tasks (threads) architecture can accommodate most real-time applications with reasonable real-time performance. For the onboard system, we have peripheral hardwares such as wireless transceivers, wi-fi card, inertial measurement unit, data acquisition board and servos. To realize coordinate control, the block of supervisor and coordinator and the control block are also defined as subtasks onboard. The hierarchical architecture of onboard tasks composed of five layers is illustrated in Fig. 4.

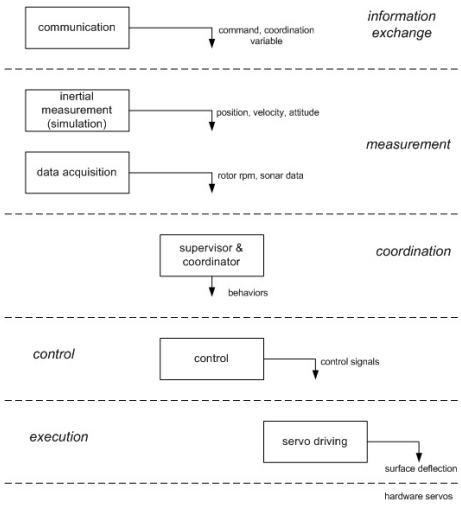

Fig. 4. Architecture of onboard tasks.

The communication module realizes the information exchange among UAVs and status data feedback to GCS. And information exchange among UAVs can be selected as centralized or decentralized. Also the communication block can respond to user commands to perform tasks. The commands received by the communication block is parsed and dispatched into the corresponding behaviors with the properly designed control law. The communication module incorporates the mechanisms of both serial wireless and the wi-fi. Furthermore, the wi-fi is implemented in a client-server architecture, where client task is mainly for sending query coordination data and server task is mainly for sending replied coordination data.

The following onboard data flow comes from the output measurement which consists of IMU and data acquisition board (DAQ). The measurable output includes position, velocity, attitude from IMU, and rotor rpm, sonar height readings from DAQ. Since the altitude from GPS has a relative large error, the sonar is used to obtain accurate height data to assist height control such as take-off and landing. On the other hand, to facilitate indoor simulation, another simulation block is developed to substitute the measurement output from IMU. The simulation is based on the UAV model derived in the procedure of system identification.

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 itself status to derive the coordination behavior according to the identity of the UAV. For example, in a centralized form of formation flight, the supervisor in the leader determines the state transition based on the input from its coordinator. In the followers, the coordinator will receive the task assigned by the leader while the supervisor does nothing.

With the updated status and dispatched coordinate behavior, the control system block will be activated to derive the control signal output for the execution block.

The execution block refers to the servo driving. It will feed the output of controller to the servos (including aileron, elevator, auxiliary and rudder) to drive the surface deflections to the desired positions. And the correct behaviors will be executed such as fly forward, hover, head turning and etc.

2) Onboard Task Management: The onboard system on each UAV is the primary component to realize distributed, cooperative behavior. It consists of the following tasks: sensor information retrieval, cooperative information processing, control algorithms computation, servo driving execution, coordination and control, client and server, wireless communication and data logging. The onboard tasks run in a multi-thread manner and task thread architecture is shown in Fig. 4. The task sequences are executed in the following order: IMU and DAQ are executed first to retrieve the sensor information, then SVR receives the coordination data, then CTL algorithms including coordination and supervision are calculated, and the output control signals are dispatched to the servos SVO, followed by data communication CMM and CLT and data logging DLG. The execution of all the task threads are managed by the onboard main program. The overall tasks are coordinated to run like a lotus one by one. The onboard real-time system is implemented on the QNX Neutrino RTOS 6.3.2, which suits for the embedded real-time applications. It provides multitasking, threads, priority-driven preemptive scheduling,

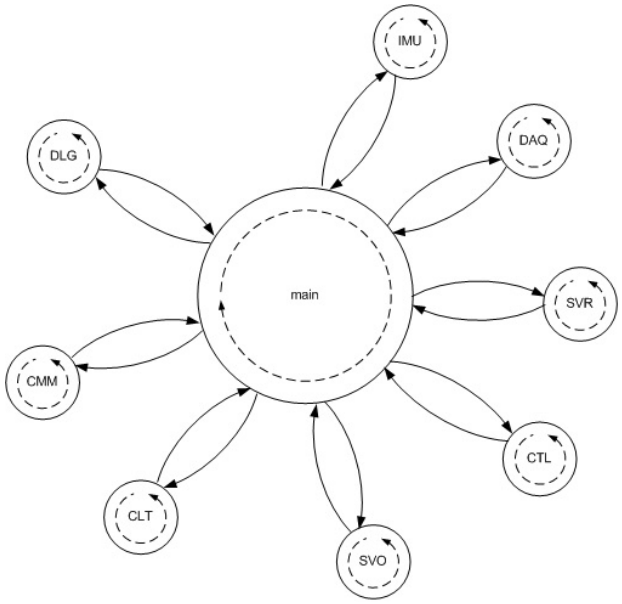

Fig. 5. Onboard task management.

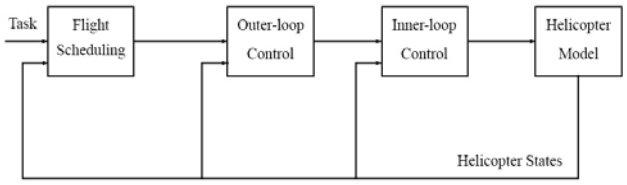

Fig. 6. The architecture of the control system.

and fast context-switching [8]. The task management is shown in Fig. 5.

We verify our software system in the formation flight cooperative scenario. Cooperative formation flight control is implemented based on the behavior-based control architecture on each UAV. The control architecture has two parts, the task scheduling and the control system. The task scheduling is to generate a sequence of behaviors from user commands and environment information, while the control system is for executions of behaviors. Interested reader may find details in [7]. The overall control architecture is shown in Fig. 6. It is composed 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 velocity 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. The task of the inner-loop controller is to follow the output of the outer-loop controller.

B. Ground Control Station

In this multiple-UAV system, the GCS is capable of monitoring and commanding one individual UAV or the fleet of UAVs. In the flight tests, the status information of all the UAVs are transferred to the GCS and displayed in the GCS. The flight status are shown in different visual perspectives. One great feature of the GCS is that the cooperative paths of the fleet can be demonstrated in a Google Map view.

The software system on the GCS is realized with MFC (Microsoft Foundation Class) in a laptop with the Windows XP Professional operating system. The overall architecture is realized via the MDI (Multiple Document Interface) approach, as shown in Fig. 7. We integrate several visual perspectives for the demonstration of flight status data from multiple UAVs. The document class is for the management of data sending and receiving, and periodic update of all the views consisting of status view, command window, gauge view, curve view and Google Map waypoint view.

The GCS is composed of background tasks and foreground tasks. 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. Consequently, the corresponding multiple display in the foreground layer is executed, and the cooperative waypoints of the paths are demonstrated. Similarly, the upload link can broadcast the commands to all UAVs, or alternatively send command to a specific UAV, both via the sending task. The global data are dynamically updated from the background layer.

Specifically, based on our previous development for single UAV, we incorporate the Google Map view to better demonstrate the cooperative behaviors of the fleet of multiple UAVs. We capture the map from Google Earth where we will conduct outdoor flight test and record the GPS data on the corners of the map. 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 waypoint view. For indoor flight test, since the GPS signal is not available, we utilize the relative position information to simulate this functionality in the Google Map view.

IV. Results of Indoor Simulation and Outdoor Flight Test

A. Indoor Simulation

Indoor simulation is useful to test the overall behaviors of both onboard systems and the GCS. With a built-in UAV model, we can conduct the hardware-in-the-loop simulation. In such simulation, the interactions among multiple UAVs and GCS and precautions under different failure situations are tested so that we can guarantee the UAV can render predictable performances in the outdoor flight.

1) Simulation Scenario: In the formation flight, the leader is commanded to perform a predefined path tracking, and the task of the follower is to follow the leader with a 10 m distance offset in the axis of longitudinal and lateral in the coordinate of the leader. In this two-UAV cooperative situation, HeLion is assigned as the leader, while SheLion is the follower. The final overall formation flight scenario with

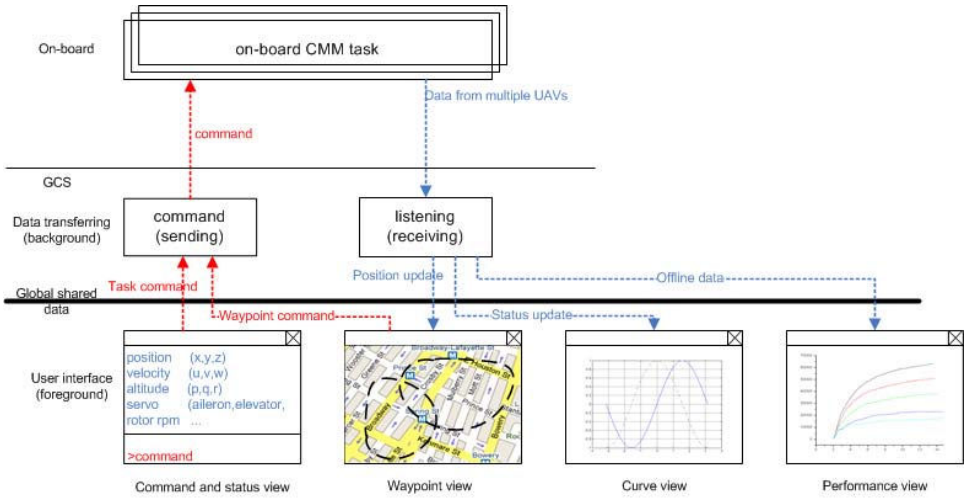

Fig. 7. Architecture of the ground control station.

a circle path is demonstrated in Fig. 8, where L_0 and F_0 are the initial reference rendezvous positions for the leader and the follower respectively. The points L_i $(i = 1, 2, ..., N)$ refer to the predefined trajectory for the leader, and the points F_i $(i = 1, 2, ..., N)$ refer to the reference points that the follower receives from the coordinator of the GCS.

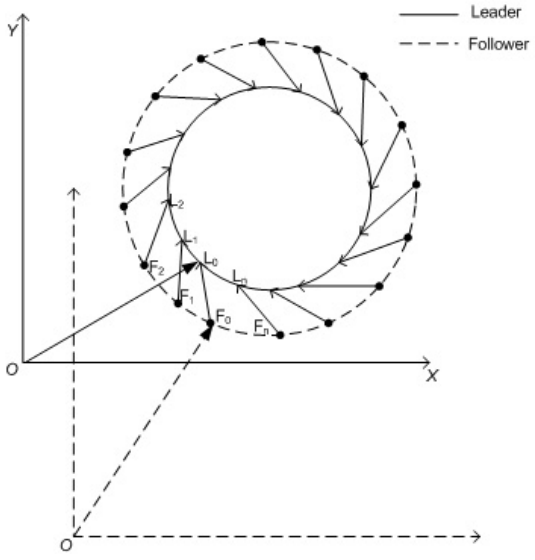

Fig. 8. Leader-follower circle formation scenario.

2) Simulation Results: In the simulation, the initial starting point of the leader and follower can be determined on the Google Map. And the information exchange frequency is set to 2 updates per second in the GCS. In this simulation, the leader is commanded to perform a circle path tracking with a radius of 10 m with an average velocity of 1 m/s. The simulation results in the GCS can be seen in Fig. 9.

In addition, Fig. 10 and Fig. 11 show the performance of the formation flight. It is obvious that in Fig. 10, the follower

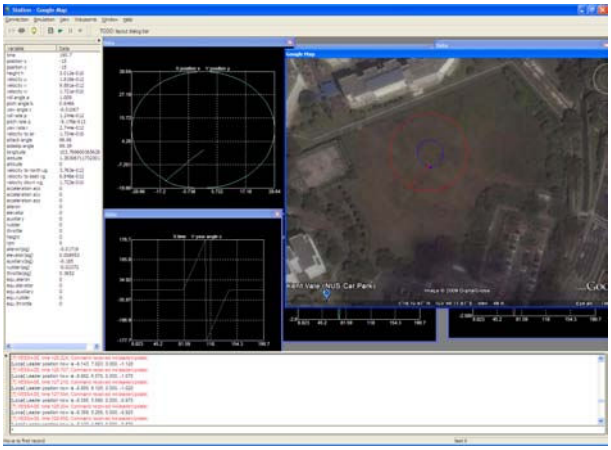

Fig. 9. Leader-follower formation in the GCS.

tracks the reference path given from the GCS correctly and timely. The tracking error between the trajectory of the follower and its reference is due to the inherent delay in tracking control. Another point is 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. On the other hand, the heading angle tracking is quite accurate despite the delay.

B. Outdoor Flight Test Results

In the outdoor flight, we conduct the line path based formation flight at first. In this scenario, the leader performs a 30 m line path tracking with an average velocity of 1 m/s. And the separation distance is set to 15 m for safety. The flight test is shown in Fig. 12 and Fig. 13.

It can be seen that the basic line path formation flight is achieved with our developed software system. It should be noted that the small fluctuations in the yaw angle comes from the inherent inaccuracy in the inertial naviga-

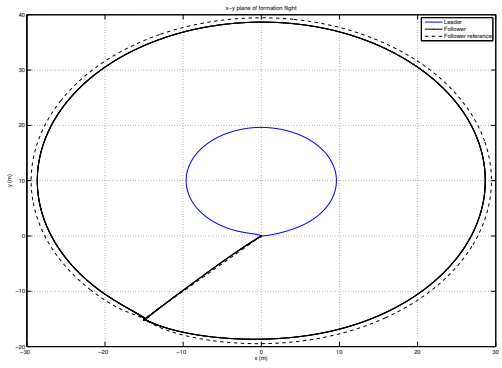

Fig. 10. Leader-follower in a circle formation - trajectory.

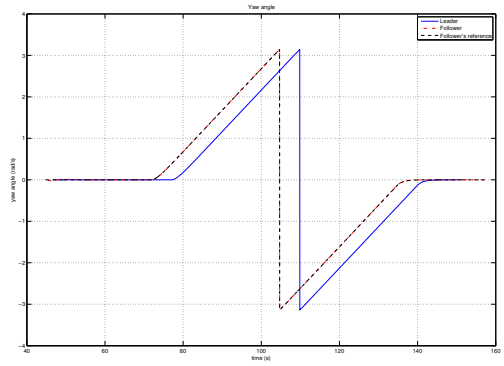

Fig. 11. Leader-follower in a circle formation - heading angle.

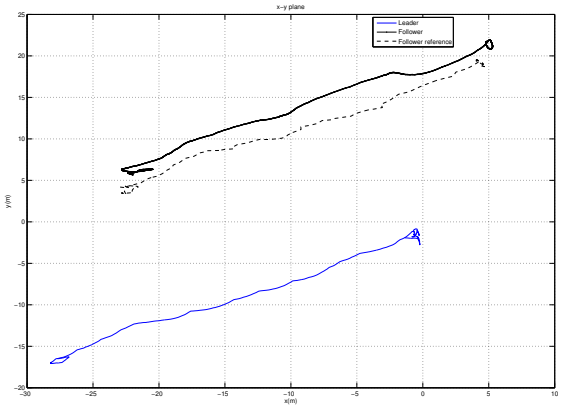

Fig. 12. Leader-follower in a line formation - trajectory.

tion system. Interested readers may visit our video link at: http://uav.ece.nus.edu.sg/∼uav/videos.htm.

V. Conclusion

The onboard software architecture fulfills the need of realizing real-time cooperative behaviors among multiple UAVs.

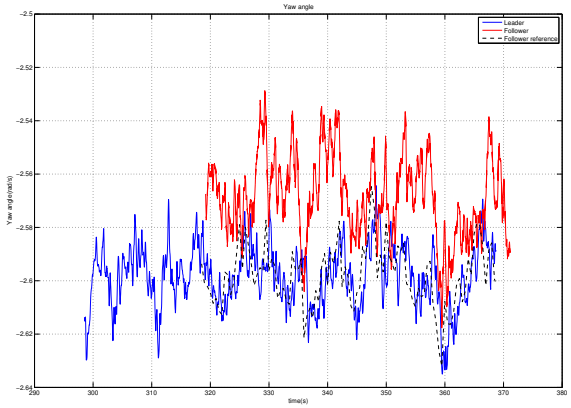

Fig. 13. Leader-follower in a line formation - heading angle.

In addition, the ground control station also meets the need of monitoring and commanding multiple UAVs. Finally, simulation of leader follower based formation is given and the outdoor experiments are performed to further verify our system performance. The software system is proved to be efficient to fulfill the real-time requirements in the multiple-UAV system.

Our future work includes the formation flight with automatic take off and landing, split and merge in case of obstacles and performance enhancement. One possible approach is to deploy the differential GPS (DGPS) to obtain more accurate position and heading angle data.

References

[1] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," IEEE Transactions on Control Systems Technology, vol. 9, no. 6, pp. 777 C 790, Nov. 2001.

[2] J. S. Jang and C. J. Tomlin, "Design and implementation of a low cost, hierarchical and modular avionics – architecture for the Dragonfly UAVs," *Proceedings of AIAA Guidance, Navigation, and Control Conference*, Monterey, California, pp. 4465-4477, 2002.

[3] J. Tisdale, A. Ryan, M. Zennaro etc. "The software architecture of the Berkeley UAV platform," *Proceedings of the 2006 IEEE International Conference on Control Applications*, Munich, Germany, 2006.

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

[5] G. Cai, K. Peng, B. M. Chen,and T. H. Lee, "Design and Assembling of a UAV Helicopter System," *Proceedings of the 5th International Conference on Control & Automation*, Budapest, Hungary, pp. 697-702, 2005.

[6] B. D. O. Anderson, C. Yu and F. Baris, "Information Architecture and Control Design for Rigid Formations," *Proceedings of the 26th Chinese Control Conference*, Zhangjiajie, China, 2007.

[7] M. Dong, B. M. Chen, G. Cai and K. Peng, "Development of a Real-time Onboard and Ground Station Software System for UAV Helicopter", *Journal of Aerospace Computing, Information and Communication*, Vol. 4, pp. 933-955, 2007.

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

[9] A. Attoui, *Real-Time and Multi-Agent Systems*, Springer, New York, 2000.