A Modular Software Infrastructure for Distributed Control of Collaborating UAVs

Allison Ryan,* Xiao Xiao,* Sivakumar Rathinam,* John Tisdale,* Marco Zennaro,*
Derek Caveney †, Raja Sengupta ‡, J. Karl Hedrick §
University of California Berkeley, Berkeley, CA, 94720, United States

Collaborating unmanned aerial vehicles can efficiently perform surveillance, mapping, and other tasks without human risk. Currently deployed unmanned aerial vehicles demonstrate a need for increased autonomy and cooperation. We present a software architecture and UAV hardware platform that have demonstrated single-user control of a fleet of aircraft, distributed task assignment, and vision-based navigation. This is the only such system known to the authors to have demonstrated in-the-air task allocation and collaboration.

A modular software infrastructure has been developed to coordinate distributed control, communications, and vision-based control. Along with the onboard control architecture, a set of user interfaces has been developed to allow a single user to efficiently control the fleet of aircraft. Distributed and vision-based control are enabled by powerful onboard computing capability and an aircraft-to-aircraft ad-hoc wireless network. Custom modifications to the Sig Rascal airframe were required to support this capability.

We describe original elements of the system that provide unique capabilities for collaboration, followed by results of a milestone flight demonstration of three collaborating UAVs.

Nomenclature

V	Airspeed
ψ	Heading angle
x, y	Position components
W_x, W_y	Wind disturbance components
$\dot{\psi}_{max}$	Turn rate constraint
u	Control input

I. Introduction

Unmanned aerial vehicles (UAVs) are a rapidly-evolving technology with a variety of civil and military applications including surveillance, search missions and combat. The United States military has deployed over 700 UAVs in Iraq,[1] and the U.S. Customs and Border Protection agency has incorporated UAVs into Phase II of the Arizona Border Patrol Initiative.[2] These types of applications can be accomplished most effectively by cooperating teams of autonomous agents.

Techniques for UAV collaboration are being developed in academic, commercial, and military settings. Ad hoc wireless networks have been implemented on UAV teams and shown to be beneficial for maintaining connectivity.[3] UAVs in cooperation with ground robots have been used to perform efficient vision-based

*Graduate student researcher, Center for the Collaborative Control of Unmanned Vehicles
†Postdoctoral researcher, Center for the Collaborative Control of Unmanned Vehicles
‡Assistant Professor, Civil and Environmental Engineering, 707 Davis Hall
§James Marshall Wells Professor, Mechanical Engineering, 5104 Etcheverry Hall, AIAA member

feature localization.[4] UAV cooperation has been addressed and examined for stability properties via distributed model predictive control[5] and receding horizon control,[6] and safe autonomous navigation for a single UAV has been studied extensively[7].[8]

Our primary contribution is the accomplishment of a major milestone in UAV cooperation: decentralized task allocation for a dynamically changing mission, via onboard computation and direct aircraft-to-aircraft communication. Performing all of the collaboration and control processing on the aircraft leads to a truly decentralized and autonomous system, and enables the fleet of UAVs to accomplish missions out of range of the ground station. To our knowledge, we are the only university group with such capability.

We present a complete system for cooperative UAV applications with integrated capabilities including in-the-air task allocation, autonomous vision-based navigation, and single-user control. Effective information fusion and high-level control become necessary to allow a single person to operate increasingly large fleets of UAVs. An important compliment to our autonomous team of UAVs is a unique graphical interface that enables monitoring and high-level control by a single non-expert user.

Section II details the system capabilities and control using a hierarchy of user interfaces. Section III describes the software architecture for collaborative control, as well as the algorithms for vision-based navigation. Section IV describes the experimental airframe and onboard computing, with emphasis on in-house adaptations that contribute to a flexible and robust research platform. Section V presents results of a flight demonstration, with conclusions in section VI.

II. Capabilities and user interfaces

The fleet of UAVs is controlled via the hierarchy of user interfaces shown in figure 1, resulting in extensive fault tolerance. High-level control is commanded by the "Team Control Interface", which is used to assign task-level commands to the entire fleet. The onboard computing processes on all aircraft are monitored via the "Process Monitor Interface", and each UAV's flight status is monitored via the Piccolo Operator Interface (included with the autopilot package), which can also be used to manually set autopilot-level commands. In the event that a control interface fails, it can be overridden by any interface lower on the hierarchy, providing multiple layers of fault tolerance.

The Team Control and Process Monitor interfaces were both developed to allow a single user to control and monitor a fleet of UAVs with varying membership. The Process Monitor simultaneously displays the status of individual software processes running on each UAV, and allows the user to send high-level commands to individual processes. Applications include commanding mode switches for a high-level controller and in-flight gain tuning for low-level controllers. If the Team Control interface fails or enters an unsafe command, it can be overridden by sending task commands to individual aircraft through the Process Monitor. Because the Process Monitor command structure is general, this interface requires minimal modification as the onboard architecture evolves.

The Team Control Interface enables a single user to monitor and control the fleet of UAVs via a user-friendly graphical environment suitable for non-technical operators. All UAVs and tasks are displayed on a map, which is also used to input new tasks. A list of the fleet members displays each UAV's map icon, current task, and most recent communication. The task list displays the list of assigned tasks, any of which can be canceled at any time. A section of the map area of the interface is shown in figure 2, including the locations of two UAVs (tagged POL for Polar and KOD for Kodiak), a visit task (101) and part of a border patrol task (green line).

A general format for handling multi-agent tasks is demonstrated via the border patrol task, which prompts the user to select the desired number of patrolling UAVs. The border is then divided into the desired number of individual UAV tasks. When the total number of assigned individual tasks is greater than the number of unassigned UAVs, the user is offered the choice of reducing the number of UAVs assigned to the border patrol task. In this event, the border task is automatically redivided into a smaller number of individual tasks and reassigned.

Other defined task types include GPS location visit and vision-based patrol of a linear feature, which was demonstrated with the California Aqueduct. In vision-based patrol, the UAV navigates using GPS to the approximate location of the aqueduct, detects and verifies it visually, and then switches autonomously to vision-based control.

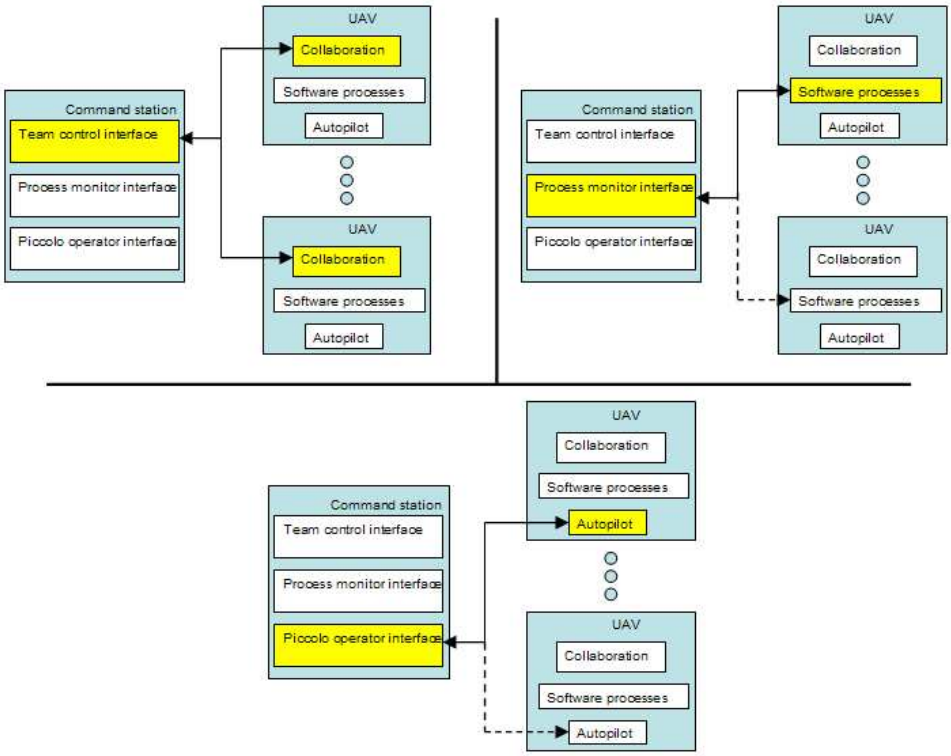

Figure 1. A hierarchy of three user interfaces allows control ranging from cooperative task assignments to individual autopilot commands. Communication from the team control interface is broadcast, while communication from other interfaces is directed to any specific UAV.

Figure 2. Part of the map area of the Team Control interface. The UAVs Polar and Kodiak are shown, as well as a visit task (101) and part of a border patrol task (diagonal line).

III. Onboard software architecture

The software architecture on each aircraft's onboard PC104 stack is based on the QNX 6.2 realtime operating system running processes including a publish and subscribe database, communications processes (via serial ports and 802.11b wireless protocol), image processing, and various controllers. The Cogent DataHub publish and subscribe database manages communication between the various processes by restricting write privileges of each shared variable to its owner process. This restriction precludes many of the difficulties associated with multi-process data management. The software architecture is shown in figure 3, which shows the main categories of processes and emphasizes the modularity of the system.

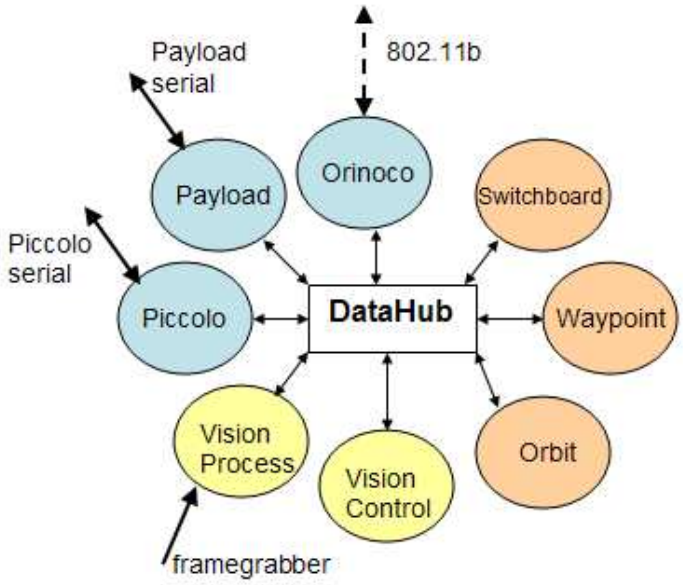

Figure 3. QNX Software architecture for collaborative UAV control. Communication processes are shown in the upper left, vision control processes are in the lower left, and other control processes are on the right. The modular architecture is easily expandable and simplifies development and debugging.

A. Essential communication infrastructure

Three communication processes run at all times: Piccolo, Payload, and Orinoco. These allow a PC104 to communicate with the Piccolo avionics, user interfaces on the ground, and the PC104s on other aircraft.

The intent of the Orinoco process communication system is to support a robust low-bandwidth communications channel between the UAVs using off-the-shelf components with minimal modification. The protocols were inspired by those used for the discovery of intradomain routers.

The aircraft-to-aircraft protocol architecture is best explained from the top down. Channel clients are high-level processes mostly making binary decisions based on low latency and low bit-rate exchanges among the UAVs. Low-latency is required for real-time decisions. Thus, communication is achieved by the periodic broadcasting of group-level state information from each vehicle. This group-state information can be visualized as a two-dimensional vector where one dimension denotes the particular vehicle and the other dimension denotes the specific field of interest such as time stamp or ID number.

Locally, the Orinoco process transport layer shares a local copy of the group-state information with the DataHub and refreshes its own entries such as GPS time and sequence number. Globally, the transport layer uses a controlled flooding technique that is used by link-state routing protocols for acquiring intradomain routing information.[9] This technique robustly disseminates all of the routing information to all of the nodes.

Controlled flooding is built into the group-state information structure in the sense that whenever a broadcast is made, the broadcaster's knowledge of other vehicles is broadcast along with its knowledge of its own state. Therefore, when an agent hears a broadcast, it will gain knowledge of not only the broadcaster

but also the broadcaster's knowledge of the other vehicles. This scheme provides robustness for information dissemination.

While controlled flooding provide a means to distribute information, a time-based aging scheme is used to discriminate useful (i.e. up-to-date) information from less useful information. Each vehicle information entry includes a GPS time stamp and a sequence number, which provide an origin and a resolution. Thus, when a broadcast is received, the protocol uses the time and sequence number of each newly received entry to decide whether or not to update the local entries. Locally, the aging of an entry diminishes its value, which is associated with a local variable called the "vote of confidence" (voc). The voc decreases every sample interval if an entry has not been updated by that particular entry's owner. It is updated only when a message is heard from the owner of that entry or from another vehicle with a higher voc number for that entry.

Together, the combination of GPS time and advancing local time provide a standard to judge the recency, and therefor value, of group state information.

The remaining communications processes, Piccolo and Payload, manage two serial connections between the PC104 and the Piccolo avionics. The Piccolo connection carries the same avionics telemetry packets that are exchanged with the Piccolo groundstation. These include aircraft telemetry such as air pressure and velocity, and commands such as waypoints. These commands are supplied by a controller supervisor process, which selects the output from whichever lower-level controller is active for the current task as described in section B. The Payload process allows data exchange between the PC104 and user interface software via the Piccolo system 900 MHz radio link. The Piccolo groundstation interface shares this data with the other two user interfaces using a client-server format.

B. Collaboration and distributed control

Collaboration occurs when a team of UAVs receives a list of user-defined tasks and autonomously assigns the tasks among themselves in a cost-effective distributed manner. The user assigns and cancels tasks in real time via the Team Control interface, with the following types of tasks currently supported.

1. Visit location
2. Patrol line
3. Vision-based aqueduct patrol

Task assignment should result in the list of tasks being completed at a low cost, where in this case cost is defined by distance traveled. A UAV calculates the cost to complete a task by computing the Euclidian distance from its current location to the point associated with the task. For example, the characteristic point associated with the aqueduct following task is the point where the UAV will first reach the aqueduct. The UAV's kinematic constraints mean that the distance it travels to reach a point is generally greater than the Euclidean distance. However, when the tasks are separated by a distance that is large compared to the UAV's turn radius, the discrepancy is small. Formally, the Euclidean distance can be considered a lower bound on the actual cost of a particular assignment.

When the task list is updated, each UAV selects the task that it could complete at the lowest cost to itself (a greedy algorithm), and then broadcasts its selection. In the event that two UAVs select the same task, a conflict is declared and the UAVs must compare reported costs. When one UAV sees that the other could complete the task at a lower cost, it "gives up" the task and selects its next lowest-cost option. When a task is canceled, the UAVs assigned to it will claim new tasks if any remain unclaimed, and otherwise will loiter at their current positions.

The three types of tasks listed above are associated with three controller processes: Waypoint, Orbit, and Vision. A controller process outputs a set of autopilot commands based on the parameters of the assigned task and the current UAV state. The Waypoint and Orbit controllers can both act on tasks made up of any number of GPS points. The Waypoint controller causes each point to be visited once, whereas the Orbit controller provides a continuous orbit of a set of points. Thus, the Waypoint controller is activated for visit location tasks and the Orbit controller is activated for patrol tasks.

A variety of higher-level behaviors can be built using these basic controllers.[10] For example, a moving vehicle can be tracked by assigning an orbit task on a point fixed on the vehicle, whose location is updated using either GPS tracking or vision. A mapping task can be accomplished by assigning many waypoints throughout the area to be mapped.

Each controller is implemented as a separate process running in QNX. A supervisor process referred to as Switchboard recognizes which controller process corresponds to the current task. The command output from that process is directed to the Piccolo process and thereby forwarded to the avionics. Although this architecture is relatively complex for dealing with such simple tasks and controllers, it provides useful modularity for future system expansion. It also abides by the policy that each variable has only one process with write permission. The system's modularity allows new controllers and processes to be developed concurrently and independently of the complete architecture, interfacing only with the publish and subscribe database.

C. Vision-based tracking of linear structures

Vision-based tasks require at least two processes: one for video processing and one for control. In this case, the video processor is the more complex of the two, and one of the main design objectives was to avoid, if possible, the use of computationally expensive algorithms. The video processing algorithm proposed in this work has two main stages. The first stage is pre-processing, where some expensive algorithms such as the Hough transform are applied before takeoff to previously collected video. The output of the pre-processing stage is a one-dimensional mean intensity profile of the image with the linear structure boundaries marked in it. The real-time stage consists of matching this mean profile with sampled horizontal stripes of the image to find the location and the scale of the linear structure. Finally a curve fitting algorithm is used to fit a cubic B-spline to the center locations. More details regarding the algorithm and its implementation are given in.[11]

An effect of placing a downward-looking camera on a UAV is that the vanishing point does not typically lie in the collected images. The effect of this is that although there is a perspective effect from the camera, the width of the linear structure changes slowly, unlike in the images collected from a ground vehicle. This aspect assists in the one-dimensional matching process because the search space reduces. Also, not every horizontal strip of the image needs to be processed. Reasonably-spaced horizontal samples are sufficient for identifying the curve of the structure.

Once a cubic B-spline has been fit to the linear structure, following it with a non-holonomic vehicle has been addressed in the literature. Autonomous tracking of curves using vision has been studied for ground vehicles by formulating the problem of following a curve using a unicycle kinematic model in the moving frame.[12] They primarily present local stability and simulation results of the control law. This control law was used for the current implementation based on its simplicity and superior performance in hardware-in-the-loop simulations compared to the other controllers investigated.

IV. Experimental platform

A research program requires a reliable yet reconfigurable experimental platform. A fleet of UAVs has been developed based on a Sig Rascal 110 airframe with extensive modifications, shown in figure 4. Low level control is provided by a commercial off the shelf autopilot, and high-level control and computation are performed on a custom PC104 stack.

A. Airframe and related hardware

The Sig Rascal 110 airframe is heavily modified due to the requirements of the PC104 payload: mainly vibration reduction and the support of extra weight. A complete payload, described in the following section, weighs as much as 6 kg, compared to the airframe's unaltered weight of approximately 5.5 kg excluding the engine. Wood and carbon fiber reinforcement throughout the body of the aircraft allow the aircraft walls to support this extra weight and improve robustness for frequent handling and disassembly.

Propulsion is provided by a front-mounted 32 cc two-stroke gasoline engine. The engine mount is designed to minimize the vibration transmitted to the airframe by using rubber vibration isolation mounts for a low-stiffness connection. This resulted in greatly improved disk drive performance relative to the standard rigid engine mount. An omnidirectional antenna for wireless ethernet is mounted on the tail of the aircraft to avoid interference from the engine and carbon fiber reinforcement. This was the passive antenna that was found to most closely approximate the radiation pattern of an isotropic emitter.

Figure 4. The modified Sig Rascal aircraft. Note wing-mounted bullet camera and payload access port above landing gear.

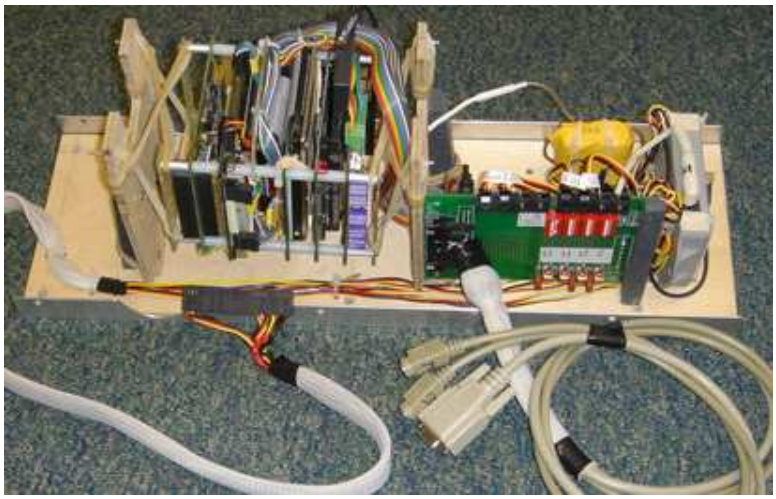

Figure 5. Payload tray with PC104, video transmitters, and power and video distribution board.

B. Payload tray

The payload tray shown in figure 5 forms the underside of the aircraft and supports the PC104 stack, video transmitters, bi-directional 1-Watt communication amplifier, and power and signal distribution system.

The PC104 computer stack includes a 700 MHz Pentium III processor, 20 GB hard disk, Orinoco Classic Gold PCMCIA wireless card, and two video frame grabbers. Although the hard disk requires extensive vibration isolation, it allows large video files to be recorded in flight, thereby avoiding the loss of quality associated with wireless transmission. These images are then used for design and simulation of vision-based navigation algorithms. It can be seen in figure 5 that the PC104 stack is suspended by latex bands from a frame built into the payload tray. This low stiffness connection is necessary for the hard disk to function properly in spite of engine vibrations. In addition, the power and data connections between the PC104 and the airframe are arranged to minimize vibration transmission through cabling.

The connections between the PC104 and the airframe include 12 volt power, two video, and two serial connections. These connections must be robust to vibration, yet frequently disconnected without damage or error. The solution is to combine all connections between the PC104 and airframe into a single umbilical shown in the lower left of figure 5. Similarly, user interface connections (keyboard, mouse, and monitor) are combined into a single exterior cable shown in the lower right. As an added benefit, pins on the PC104 are not stressed by cable torque or frequent disconnection because they interface only with the payload tray. Reliability in the field is significantly increased by eliminating the possibility of errors such as switched video connections and loosened pins.

The custom distribution board on the right side of figure 5 distributes power to the various PC104 components, video transmission amplifiers, and the one-Watt 802.11b amplifier. It also routes the video signal between the two onboard video cameras and the frame grabbers, and provides simple reconfiguration for power and data distribution.

C. Piccolo system

The Piccolo avionics system is provided by Cloud Cap Technology[13] and consists of a central controller (groundstation) and onboard Piccolo autopilots. A single user can control a number of Piccolo-equipped UAVs using the groundstation and accompanying operator interface software. The operator interface displays telemetry from each UAV and sends high-level commands to the autopilots over a 900 MHz radio link. This link is also used by the team control and process monitor interfaces to forward data to and from processes on the PC104. The autopilot can also be commanded by processes on the onboard PC104 via the Piccolo serial connection, as was described in section III.

The Piccolo autopilot controls the UAV's non-linear dynamics using nested PID loops to produce a desired airspeed, altitude, and heading rate, or to track a series of waypoints. With its non-linear dynamics controlled by the Piccolo avionics, the UAV can be represented by the following kinematic model (1) for the purpose of high-level controller design for constant-altitude applications. V is the UAV's airspeed, ψ is its heading angle, u is the control input, and W is a wind disturbance. The heading rate is limited by maximum turn rate $\dot{\psi}_{max}$.

$$\begin{aligned} \dot{x} &= V\cos\psi + W_x \\ \dot{y} &= V\sin\psi + W_y \\ \dot{\psi} &= u \\ |\dot{\psi}| &\leq \dot{\psi}_{max} \end{aligned} \tag{1}$$

V. Flight demonstration

A flight demonstration with three UAVs was performed at Crow's Landing Naval Auxiliary Landing Field, in Patterson, California in July 2005. Once airborne, the three UAVs were controlled by a single user via the map-based graphical interface described in section II. The user assigned tasks of each type and added and deleted tasks, resulting in dynamic reassignment and conflict resolution. The task of autonomously detecting and tracking the California Aqueduct was demonstrated repeatedly.

The fleet of UAVs reliably demonstrates the goals of collaborative control and autonomous navigation. Collaboration depends on communication, which was based on wireless ethernet cards and omnidirectional

antennas. Orientation proved to be a factor in connection reliability, possibly due to interference from the aircraft engine and carbon-fiber reinforcement. Contact was maintained over a range of 0.5 to 1.5 kilometers with intermittent packet loss.

A series of nine tasks was assigned to the three UAVs, including multiple-agent tasks such as cooperative border patrol. On three occasions, conflicts were declared when two UAVs claimed the same task. These conflicts were resolved, although in one case there was delay due to interrupted communication.

A UAV attempted to visually track the California aqueduct six times after the aqueduct following task was assigned. In five of the six attempts, the aqueduct was autonomously detected and tracked. The task was to track the aqueduct for approximately 700 meters before returning to the starting point.

Test video of the aqueduct was collected by flying the vehicle using waypoint navigation, and was then used to train the video processing algorithm. The airplane was held at a constant altitude of 100 meters. Sample images of the recorded video of one of the runs with vision-in-the-loop control is shown in figure 6. The cross track deviation error from the centerline of the aqueduct is shown in figure 7 . On average the deviation was approximately 10 meters over a stretch of 700 meters of the aqueduct.

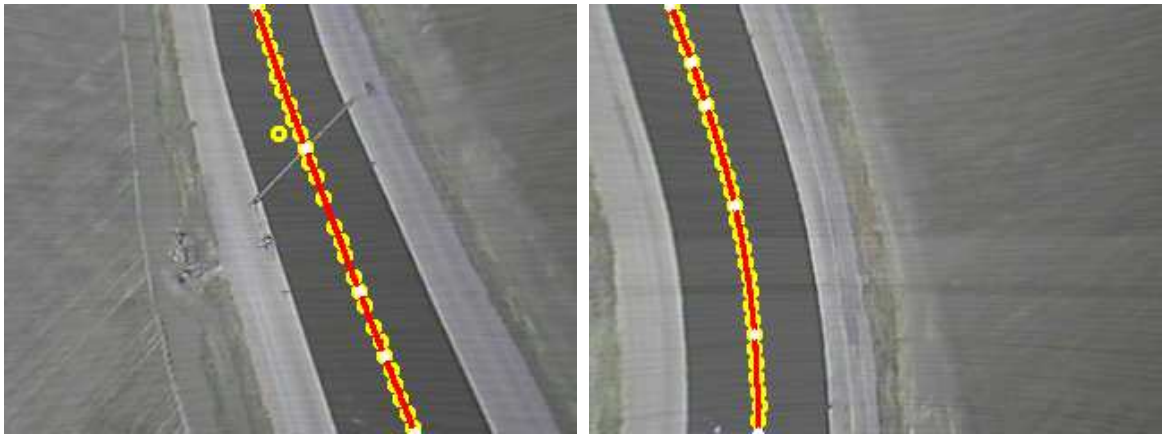

Figure 6. Sample images from the onboard processed video showing detection of the center of the aqueduct.

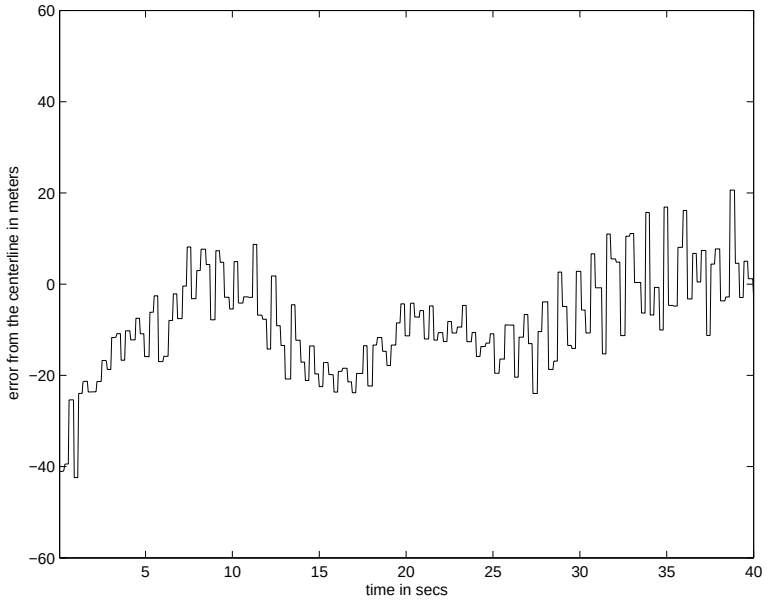

Figure 7. Cross track error from the centerline of the aqueduct.

VI. Conclusions

We have demonstrated UAV collaboration through in-the-air task allocation and conflict resolution. A single user controls the fleet of UAVs by assigning high-level tasks through a map-based interface, allowing efficient control of large numbers of aircraft. Each UAV has onboard software processes that provide low-level control, task selection and negotiation, vision-based control, and aircraft-to-aircraft communication. This combination allows a high degree of autonomy at both the individual and group level, requiring a minimum of human intervention.

Along with advances in multi-agent collaboration, a valuable experimental platform for collaborative control of UAVs has been developed. Off-the-shelf components have been extensively modified to provide a small aircraft with unusually powerful computing and communications resources, while maintaining modularity, robustness, and flexibility for new applications.

Future work includes development of a similar platform based on a larger airframe. This will provide greater weight and power allowances to allow longer flights and support a variety of sensing and communication devices. The selection of tasks will be increased to include more abstract behaviors that can be built up from the current simple controllers.

Acknowledgments

This work was supported by the Office of Naval Research. The authors would like to thank the following people for their part in hardware development and flight testing: Tim McGee, Aram Soghikian, Stephen Spry, Mark Godwin, Stephen Jackson, David Nguyen, Dan Prull, and Dan Coatta.

References

[1] "Remotely Controlled Aircraft Crowd Dangerous Iraqi and Afghan Skies," *The New York Times*, april 2005.

[2] Press release, United States Customs and Border Protection Agency, August 2005.

[3] Brown, T., Doshi, S., Jadhav, S., and Himmelstein, J., "Test Bed for a Wireless Network on Small UAVs," *proceedings of AIAA 3rd Unmanned Unlimited Technical Conference*, September 2004, pp. 20–23.

[4] Grocholsky, B., Bayraktar, S., Kumar, V., Taylor, C., and Pappas, G., "Synergies in Feature Localization by Air-Ground Robot Teams," *9th International Symposium on Experimental Robotics (ISER'04)*, 2004.

[5] Richards, A. and How, J., "Decentralized Model Predictive Control of Cooperating UAVs," *Proceedings of the IEEE Conference on decision and control*, December 2004.

[6] Franco, E., Parisini, T., and Polycarpou, M. M., "Cooperative Control of Distributed Agents with Nonlinear Dynamics and Delayed Information Exchange: a Stabilizing Receding-Horizon Approach," *proceedings of the IEEE conference on decision and control*, 2005.

[7] McGee, T., Sengupta, R., and Hedrick, J. K., "Obstacle detection for small autonomous aircraft using sky segmentation," *Proceedings of the IEEE International conference on robotics and automation*, 2005.

[8] Frew, E. and Sengupta, R., "Obstacle avoidance with sensor uncertainty for small aircraft," *Proceedings of the IEEE conference on decision and control*, December 2004.

[9] Peterson, L. L. and Davie, B. S., *Computer Networks, A System's Approach*, Morgan Kaufmann, 3rd ed.

[10] Caveney, D. and Sengupta, R., "Architecture and Application Abstractions for Multi-Agent Collaboration Projects," *proceedings of the IEEE conference on decision and control*, December 2005.

[11] Rathinam, S., Kim, Z., Soghikian, A., and Sengupta, R., "Vision Based Following of Locally Linear Structures using an Unmanned Aerial Vehicle," *proceedings of the IEEE conference on decision and control*, 2005.

[12] Frezza, R., Picci, G., and Soatto, S., *The confluence of vision and control*, chap. A lagrangian formulation of nonholonomic path following, Springer Verlag, 1998.

[13] Vaglienti, B., Hoag, R., and Niculescu, M., "Piccolo system user guide," www.cloudcaptech.com.