ELSEVIER

Annual Reviews in Control 29 (2005) 101–115

Annual Reviews in Control

www.elsevier.com/locate/arcontrol

From mission planning to flight control of unmanned aerial vehicles: Strategies and implementation tools☆

George Vachtsevanos *, Liang Tang, Graham Drozeski, Luis Gutierrez

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250, USA

Received 18 August 2004; received in revised form 15 November 2004; accepted 16 November 2004
Available online 11 April 2005

Abstract

This paper reviews aspects of unmanned aerial vehicle (UAV) autonomy as suggested by the Autonomous Control Logic chart of the U.S. DoD UAV autonomy roadmap; levels of vehicle autonomy addressed through intelligent control practices and a hierarchical/intelligent control architecture are presented for UAVs. Basic modules of the control hierarchy and their enabling technologies are reviewed; of special interest, from an intelligent control perspective, are the middle and high echelons of the hierarchy. Here, mission planning, trajectory generation and vehicle navigation routines are proposed for the highest level. At the middle level, the control role is portrayed by mode transitioning, envelope protection, real-time adaptation and fault detection/control reconfiguration algorithms which are intended to safeguard the UAV's integrity in the event of component failures, extreme operating conditions or external disturbances. The UAV thus exhibits attributes of robustness and operational reliability assuring a satisfactory degree of autonomy. The control technologies are demonstrated through flight testing results.

Keywords: Intelligent control; Autonomous vehicle; Flight control; Adaptive control; Fault detection; Fault tolerant control

1. Introduction

Recent world events have highlighted the utility of unmanned aerial vehicles (UAVs) for both military and potential civilian applications. However, the reliability of these systems has been disappointing in practice. According to a recent report, nearly half of the current-generation unmanned surveillance aircraft built have been lost. This loss-rate is about 10 times worse than manned combat aircraft. Clearly these numbers are driven in part by the dangerous missions these aircraft are tasked with, but there are other factors at work here. In manned aircraft, the pilot functions as the central integrator of the onboard subsystems and works to mitigate problems when they occur. Although ''human error'' is attributed as the most common cause of aviation accidents, human pilots are also simultaneously the most important safety-enhancing component on a manned aircraft.

To address this and other related UAV control issues, the Defense Advanced Research Projects Agency (DARPA) and the U.S. Air Force Research Laboratory (AFRL) have launched a major initiative to develop revolutionary new Software Enabled Control (SEC) systems with applications to intelligent UAVs (DARPA, 2004).

Beyond the responsibility of responding to unexpected system faults, the SEC program is also charged with making these machines more agile, thus helping them avoid hostile actions without exceeding critical flight parameters. This has the potential to improve the loss-rate for even the most dangerous missions.

The SEC program includes 16 organizations divided into SEC technology developers of control-related algorithms and SEC developers of the software infrastructure platform that enables the design and implementation of advanced control methods. The organizations include: Boeing Phantom Works, University of California at Berkeley, California Institute of

☆ An earlier version of this paper was presented at the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV 2004), 5–7 July 2004, Lisbon, Portugal.

* Corresponding author.
E-mail address: giv@ee.gatech.edu (G. Vachtsevanos).

1367-5788/$ – see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.arcontrol.2004.11.002

Technology, the Charles Stark Draper Laboratory, Cornell University, Georgia Institute of Technology, Honeywell Laboratories, Massachusetts Institute of Technology, Northrop-Grumman Corporation, University of Minnesota, Oregon Research Institute, Rockwell Science Center, Stanford University, Stanford Research Institute, Scientific Systems Company, Inc., and Vanderbilt University.

Improved performance of UAVs is expected to be achieved when such vehicles are endowed with levels of autonomy that will allow them to operate safely and robustly under external and internal disturbances, to be able to accommodate fault conditions without significant degradation of their performance, to adapt to unexpected events and to coordinate/cooperate among themselves to accomplish mission objectives. Fig. 1 depicts the expected UAV autonomy capabilities according to the U.S. DoD's UAV autonomy roadmap (DoD, 2002).

In this paper, we suggest the hardware, software and control technologies aimed to achieve such autonomy objectives. The paper has the following structure: we begin in Section 2 with a description of the overall mission intelligence flow. Mission planning and a modified A* search algorithm for route planning are described in Section 3. The hardware, software and avionics configuration of our test vehicle, the Georgia Tech RMAX helicopter is presented in Section 4. Section 5 introduces the Open Control Platform (OCP), a real-time middleware platform that enables the development, demonstration and evaluation of advanced UAV control systems. In Sections 6 and 7, we focused on two control technologies, namely the adaptive mode transition control (AMTC) and fault tolerant control. Detailed controller architecture and algorithms design information is described and flight test results are presented. Conclusion and remarks on future research directions are given in Section 8.

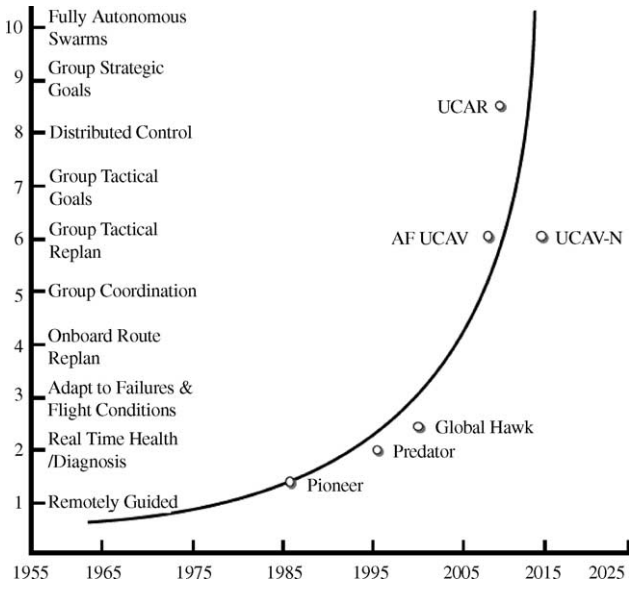

Fig. 1. Autonomous control level trend.

2. Mission intelligence flow

A hierarchical control structure for mission intelligence flow is illustrated in Fig. 2. Situation awareness is used for mission planning and flight mode selection, which constitutes the high level control elements. For inhabited aircraft the pilot and other crewmembers provide the intelligence for interpreting the data from a variety of sources to execute these functions. Much of this data is used in pre-flight or pre-mission planning and is updated onboard as the mission proceeds. As the mission segments are executed and abnormal events are encountered, flight mode switching takes place which constitutes the mid level control element. On an inhabited aircraft the pilot flies the aircraft and makes necessary mode switching and control reconfiguration decisions for implementation through the use of the flight control system. This constitutes the low-level control element and is used to execute the smooth transition between modes of flight, i.e., transition from hover or take-off to level flight, etc., and stay within the flight envelope of the UAVs. External abnormal conditions cause the pilot to take corrective action, such as avoiding an obstacle or evading a target or threat. Internal abnormal conditions can also occur, such as a failure or malfunction of a component onboard the aircraft. Once again the pilot provides the intelligence to take the corrective action by reconfiguring his/her set of controls to safely continue to fly or land the aircraft.

Without a pilot onboard the aircraft a UAV must either be controlled from the ground by a radio control ground pilot or the UAV must have its own intelligence to fly autonomously. Executing a vertical take-off and landing (VTOL) UAV mission autonomously has been demonstrated by both the Georgia Tech and Sikorsky Aircraft UAVs in the Army's Advanced Scout Rotorcraft Testbed (ASRT) Project (Schrage et al., 1997). However, both of the aircraft were not able to use the entire flight envelope capability of the UAVs, largely limited by the control algorithms imple-

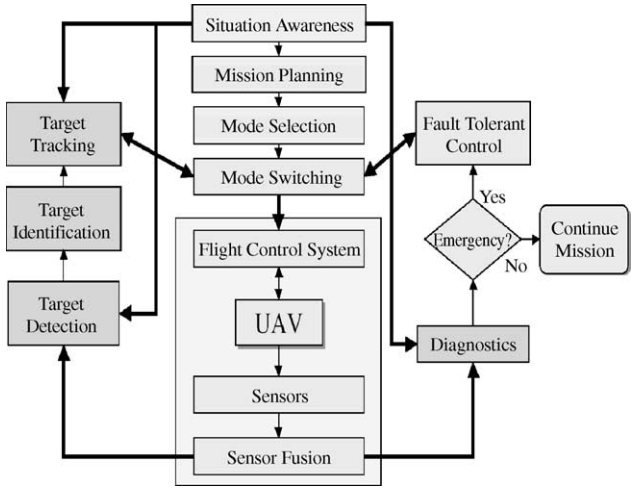

Fig. 2. Mission intelligence flow.

mented. In addition, the control algorithms were very much customized for the particular vehicle's characteristics and were developed in very much of a trial and error approach. Also, the computing architecture onboard the aircraft did not provide the environment for reusability and reconfigurability, let alone for plug and play of different SEC algorithms (Schrage & Vachtsevanos, 1999).

3. Mission planning

Fig. 3 depicts the configuration of the mission planner. The high level supervisory controller receives mission commands from the command and control post and decomposes them into sub-missions which will then be assigned to connected function modules. Upon reception of start and destination points from the supervisory controller, the route planner generates the "best" route in the form of waypoints for the UAV to follow. A database of the terrain in the form of a digitized map is available to the route planner (Vachtsevanos, Kim, Al-Hasan, Rufus, Simon, Schrage, et al., 1997).

The configuration of the route planner is depicted in Fig. 4. The digitized map is in the form of a mesh of equal square cells, where each cell is either free or occupied by an obstacle. Having two free cells (i.e., a start and a destination) assigned by the supervisory module, the A* search engine searches the map mesh and plans a cell-based route that extends from the start cell to the destination cell and avoids stationary obstacles. The generated cell route is suboptimal in terms of distance, safety and maneuvering (e.g., turning angles), i.e.,

$$\text{route cost} = (W_d \times \text{distance}) + (W_h \times \text{hazard}) + (W_m \times \text{maneuvering}),$$

where W_d, W_h, and W_m are weights for the three cost components, and are assigned by the supervisory module based on the objectives and circumstances of the mission.

The cost elements are expressed as fuzzy membership functions reflecting the inherent uncertainty associated with the planned trajectory, the obstacles along the path and the maneuvers the vehicle is required to perform as it navigates through the terrain. A* uses heuristic knowledge about the closeness of the goal state from the current state to guide the search. The cost of every searched cell, n, is composed of two components:

$$\text{cost}(n) = \text{kg} \times \text{g}(n) + \text{kh} \times h(n),$$

where $g(n)$ is the cost of the least-cost route (found in the search so far) from the start cell to cell n, $h(n)$ the heuristic

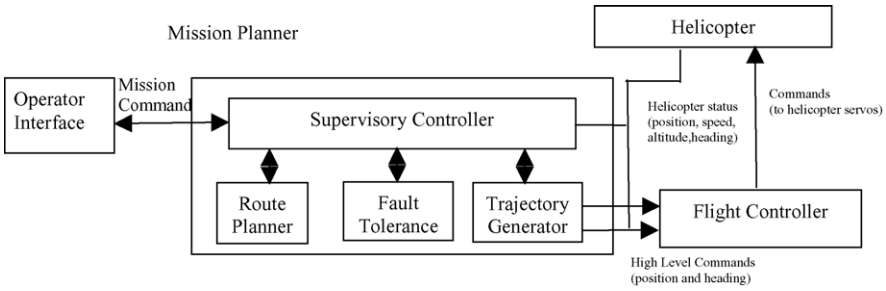

Fig. 3. The mission planning configuration.

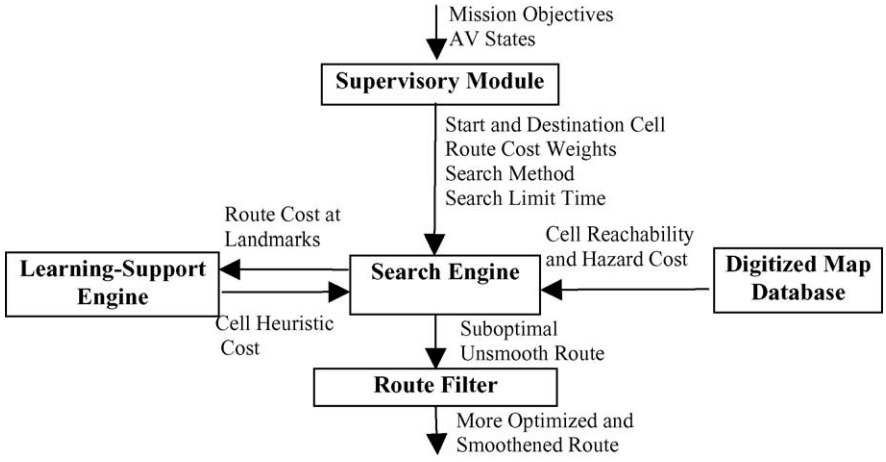

Fig. 4. The route planner configuration.

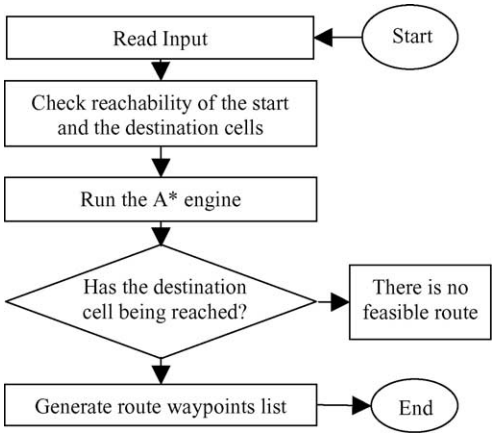

Fig. 5. Flow chart of the route planner.

(i.e., estimated) cost of the minimum-cost route from cell n to the destination cell, and kg, kh are the weighting factors for $g(n)$ and $h(n)$, respectively. Given a search state space, an initial state (start node) and final state (goal node), A* will find the optimal (least cost) path from the start node to the goal node, if such a path exists (Vachtsevanos, Kim, Al-Hasan, Rufus, & Simon, 1997). The generated cell route is further optimized and smoothed by a filtering algorithm.

The filtered route is a series of consecutive waypoints that the UAV can navigate through. The supervisory module reads the objectives and the status of the mission and based on that it configures the search engine and assigns weights to the route's three cost components. Furthermore, the supervisory module chooses the start and the destination cells for the search engine depending on the current status of the UAV, i.e., whether it is stationary or already navigating towards a destination and needs to be redirected to another destination. The learning-support module acquires route cost data from the search engine at certain map landmarks and updates a cost database that is used later to provide better heuristics to guide the search engine.

Fig. 5 illustrates in a flow chart the route planning implementation steps. Typical route planning results for an UAV with actual mapping data is shown in Fig. 6. The interested reader can refer to Al-Hasan and Vachtsevanos (2002) for more details regarding the route planner algorithms and design.

4. Flight testing: GTMax research UAV

The GTMax research UAV system (Fig. 7), developed at the Georgia Institute of Technology to support SEC and other ongoing programs, utilizes a Yamaha R-Max helicopter, a modular/open avionics system, Open Control Platform, a set of baseline onboard software, and a series of simulation tools. The baseline systems enable autonomous flight of the normally remotely piloted aircraft. The R-Max configured with these systems is known as the GTMax, a highly effective UAV research vehicle that has a design based on lessons learned from UAV research at academic institutions such as Georgia Tech, University of California at Berkeley, Massachusetts Institute of Technology, and Carnegie Mellon University for more than 10 years.

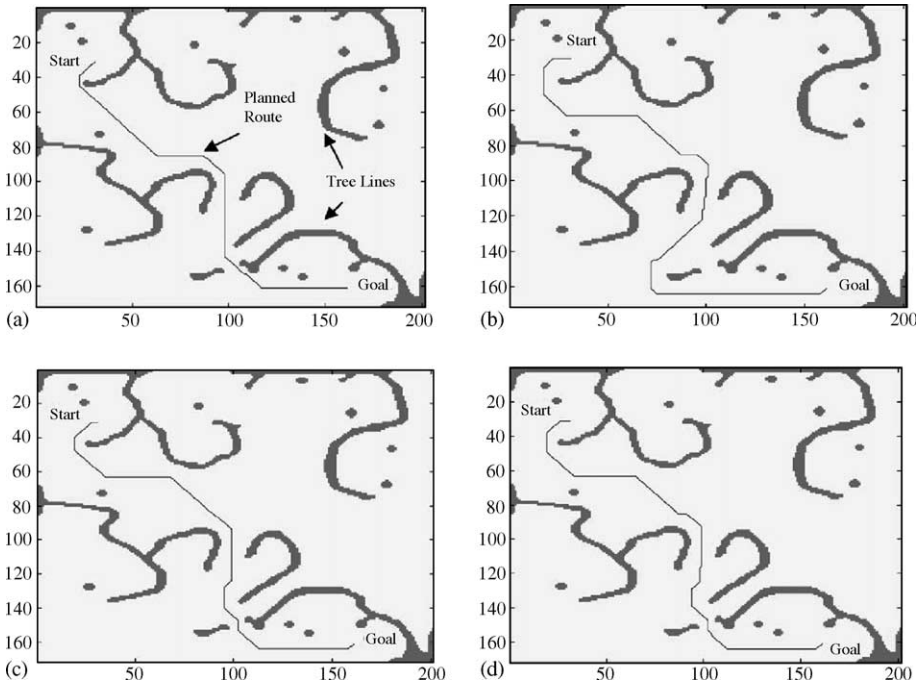

Fig. 6. Four planned unfiltered routes: (a) minimum distance; (b) minimum hazard (or maximum safety); (c) minimum maneuvering; (d) minimum distance + hazard).

Fig. 7. GTMax helicopter in flight, Georgia Tech's test bed research UAV.

Developed in Japan, the basic Yamaha R-Max helicopter has a rotor diameter of 10.2 ft, a 21 hp two-cylinder engine, and weighs 125 lb. The weight increases to 160 lb when configured with typical GTMax avionics. It is capable of carrying approximately 50 additional pounds of research equipment. It also has a generator, starter, and can be flown manually by a remote pilot in sight of the helicopter or by an onboard autopilot.

- Basic Yamaha R-Max dimensions:
  - maximum length: 3630 mm (rotor blade included);
  - fuselage length: 2750 mm;
  - width: 720 mm;
  - height: 1080 mm;
  - fuel tank: 6 L;
  - main rotor diameter: 3115 mm;
  - tail rotor diameter: 545 mm;
  - maximum gross weight: 93 gN;
  - maximum payload: 30 gN.
- Powerplant:
  - type: gasoline 2 cycles;
  - cylinder configuration: horizontal opposition 2 cylinder;
  - displacement: 246 cm^3;
  - engine RPM. 6350 RPM (nominal);
  - maximum power output: 15.4 kW (21PS);
  - maximum torque: 25.5 Nm;
  - cooling type: liquid cooling;
  - fuel: auto gas.

The GTMax avionics system hardware consists of a set of modules that can be added/removed as required for a flight test. All modules include electro-magnetic interference protection and their own power regulation. The modules are mounted in a vibration-isolated rack within an enclosure under the fuselage. The basic system includes a general-purpose computer, Differential Global Positioning System (D-GPS), an inertial measurement unit, an ultrasonic altimeter, a three-axis magnetometer, and two wireless data links. Other flight configurations used to date have also utilized a second general purpose computer, cameras, a radar altimeter, and video capture/compression hardware. The basic ground equipment includes the data links, a GPS reference station, and one or more laptop computers.

The baseline onboard software includes interfaces to the sensors, an integrated navigation filter, and a nominal trajectory-following autopilot. This allows the baseline system to fly a prescribed mission on its own, including the take-off and landing. The role of the human operator can be to set the desired flight path, start the engine, monitor the flight, and to shut down the engine after landing. This enables a large number of relevant flight control scenarios to be tested, from purely manual control to autonomous operations.

Most high and middle level control components are running on the secondary computer. The implementation of these controls modules and the interface and communication with the primary computer are based on the Open Control Platform.

5. The Open Control Platform

The Open Control Platform is being developed for use as a software platform enabling demonstration and evaluation of advanced UAV control systems technologies being developed for the DARPA Software Enabled Control program. The OCP enabling software technology is being developed by a team from industry and academia, led by Boeing Phantom Works, and including the Georgia Institute of Technology, Honeywell, Laboratories, and the University of California Berkeley. The OCP provides a middleware-base execution framework, Application Programmer Interfaces (APIs) simulation tools, and integration with useful software and control systems design and development tools. It also provides the distributed control and reconfigurable architecture that allows control and intelligence algorithms at all levels and timescales to interact in a decoupled, real-time and distributed fashion. A schematic of the required distributed control reconfigurability is illustrated in Fig. 8. As illustrated, the controller strategy interfaces with the UAV dynamic configuration, either through simulation or the hardware testbed through the OCP.

The OCP consists of multiple layers of APIs that increase in abstraction and become more domain specific at the higher layers (Guler, Clements, Wills, Heck, & Vachtsevanos, 2003), as shown in Fig. 9. At each level, the abstract interfaces are defined to provide access to the underlying functionality while hiding details of how that functionality is implemented. Each layer builds on the components defined in lower layers.

The layers of the OCP are intended to form a bridge from the controls domain to distributed computing and reconfigurability technologies so that controls engineers can exploit these technologies without being experts in computer science.

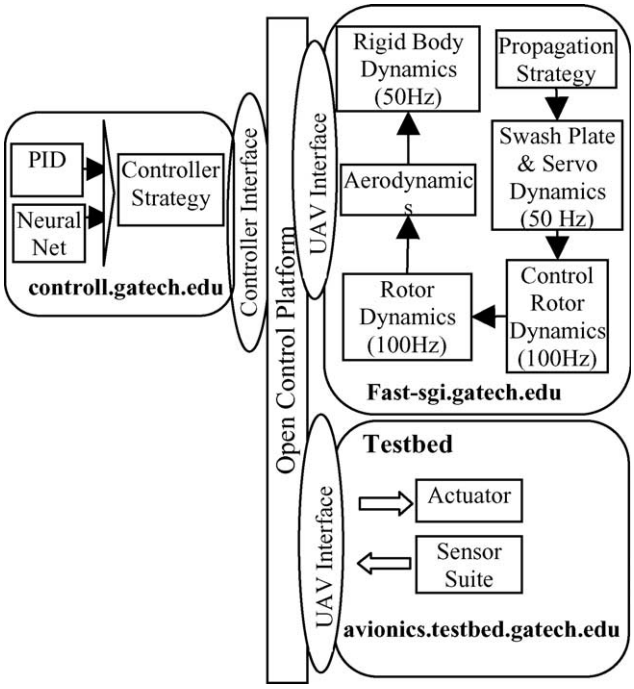

Fig. 8. Distributed control functionality.

In the bottommost ‘‘core’’ layer, the OCP leverages and extends new advances in real-time distributed object oriented computing that allow distributed components to communicate asynchronously in real time (Levine, Gill, & Schmidt, 1998; Levine, Mungee, & Schmidt, 1998; Schmidt & Kuhns, 2000; Schmidt, Levine, & Harrison, 1997). It also supports highly decoupled interaction among the distributed components of the system, which tends to localize architectural or configuration changes so that they can be made quickly and with high reliability.

The middle ‘‘reconfigurable controls’’ layer provides abstractions for integrating and reconfiguring control system components; the abstractions bridge the gap between the controls domain and the core distribution substrate (Wills, Kannan, et al., 2000; Wills, Sander, et al., 2000). The abstract interface is based on familiar control engineering concepts, such as block diagram components, input and output ports, and measurement and command signals. It allows real-time properties to be specified on signals that translate to quality-of-service (QoS) constraints in the core real-time distribution substrate. It also allows run-time changes to be made to these signal properties, which are then handled by lower-level dynamic scheduling and resource management mechanisms (Cardei, Cardei, Jha, & Pavan, 2000). This layer raises the conceptual level at which the controls engineer integrates and reconfigures complex, distributed control systems.

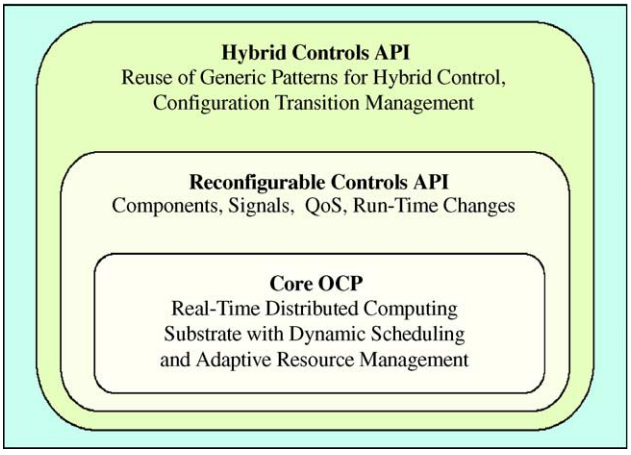

Fig. 9. Layers of the OCP.

The third ‘‘hybrid controls’’ layer supports reconfiguration management by making reconfiguration strategies and rationale for reconfiguration decisions explicit and reusable. It contains generic patterns of integration and reconfiguration that are found in hybrid, reconfigurable control systems. It can be specialized with logic for choosing reconfigurations as well as signal blending strategies for smoothly transitioning from one configuration to another. This is critical to hybrid systems in which continuous dynamics must be maintained between discrete reconfiguration events and where multiple control and blending strategies are applicable.

These aforementioned OCP capabilities and features are being delivered to the control systems technology teams on the SEC program, to provide them with a platform to enable rapid development, test and migration of advanced control systems designs to embedded software. (Wills, Kannan, et al., 2000; Wills, Sander, et al., 2000; Wills et al., 2001).

In the spring of 2002, Georgia Institute of Technology and Boeing demonstrated elements of the OCP in flight successfully. In the demonstration, using the Georgia Tech research UAV helicopter GTMax described above, the system successfully compensated for the simulated in-flight failure of a low-level flight control system by reconfiguring the software systems on its own. Since then, the OCP has become an implementation and integration platform for our UAV research activities. Flight tests also demonstrated the ability of the OCP to manage sensing, flight control algorithms and actuators to allow autonomous dynamic low-level flight control reconfiguration, as well as airborne real-time plug-and-play of external controllers developed by the SEC control developers from different organizations. The OCP is proven to represent an advance in open systems able to handle large volumes of data and computations in real time.

Illustrated in Fig. 10 at the mid-level are SEC algorithm approaches developed at Georgia Tech for implementation through the OCP: adaptive limit detection and avoidance (Yavrucuk, Unnikrishnan, & Prasad, 2003), adaptive mode transition controller, fault detection, and fault tolerant control reconfiguration, which will be discussed in following sections.

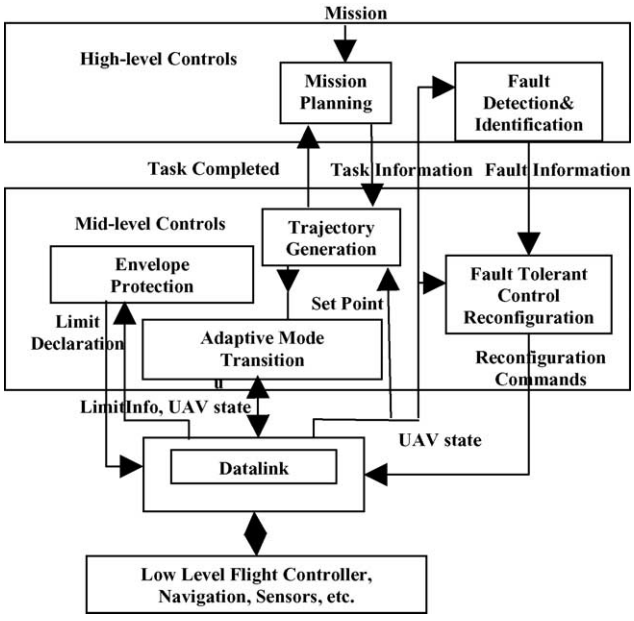

Fig. 10. Hierarchical control architecture and Open Control Platform (OCP).

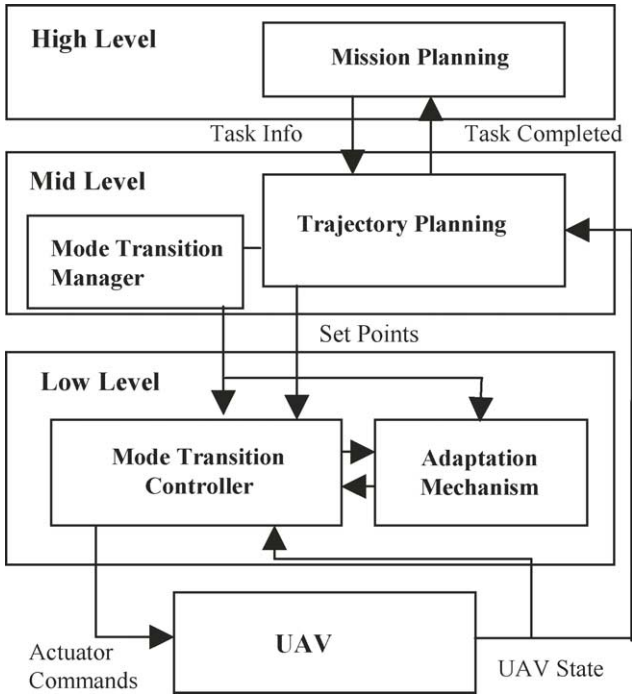

Fig. 11. AMTC hierarchical control architecture.

6. UAV adaptive mode transition control

Control of autonomous aerial vehicles presents unique challenges not only in the design of control algorithms, but also in the strategies and methodologies used to integrate and implement those algorithms on the actual vehicles. We propose an approach to the adaptive mode transition control of UAVs. The main objective of this architecture is to improve the degree of autonomy/intelligence of the UAV and its performance under uncertain conditions, for instance when external perturbations are present. The architecture is based on concepts developed in (Rufus, 2001; Rufus, Heck, & Vachtsevanos, 2000; Rufus, Vachtsevanos, & Heck, 2000, 2002) where the adaptive mode transition control scheme was first introduced. Here, we suggest a new approach to the adaptive mode transition control problem and we are introducing a hierarchical architecture to implement it. The algorithms have been implemented and tested using the Open Control Platform and validated by flight tests on GTMax (Gutierrez, Vachtsevanos, & Heck, 2003a, 2003b).

Here are some definitions used in this section: a local mode is a region of the state space around an operating point in which the vehicle exhibits quasi steady-state behavior. A local controller is a controller that guarantees the stability and tracking performance of the closed loop system for any feasible reference trajectory in a local mode. A transition region is a region of the state space outside any local mode that includes all the feasible trajectories between two local modes. The operating region is the region of the state space generated by the union of all the local modes and transition regions.

The proposed architecture for the control of UAV's consists of a hierarchy of three levels as depicted in Fig. 11. At the highest level, the mission planning component stores information about the overall mission, generates a low-level representation of that mission, and coordinates its execution with the middle level. The middle level includes a trajectory planning component, which receives information from the high level in terms of the next task to be executed to fulfill the mission, and generate the trajectory (set points) for the low-level controller. Mode transition manager (MTM) coordinates mode selection, switching and transition automatically based on the actual state of the vehicle. At the lowest level, an adaptive mode transition controller coordinates the execution of the local controllers (one for each local mode) or the active control models (one for each transition), which stabilize the vehicle and minimize the errors between the set points generated by the middle level and the actual state of the vehicle. The adaptive mode transition control consists of the mode transition control component and the adaptation mechanism component. Fig. 12 shows the structure of the proposed adaptive mode transition control (AMTC) algorithm, where C_i is the controller for local mode i.

6.1. Mode transition manager

The mode transition manager coordinates the transitions. Unlike the previous work (Rufus, 2001; Rufus, Heck, et al., 2000; Rufus, Vachtsevanos, et al., 2000; Rufus et al., 2002) where the transitions were pre-scheduled and

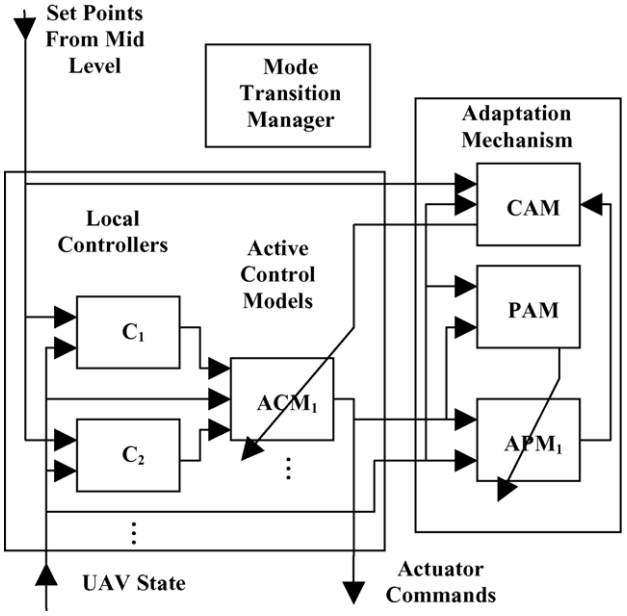

Fig. 12. Adaptive mode transition control.

a Mode Selector module coordinated the transitions, the MTM coordinates the transitions automatically based on the actual state of the vehicle. In order to accomplish this task, a Mode Membership Function is defined for each local mode and the MTM determines which local mode or transition should be activated relying upon these constructs.

For local mode i the Mode Membership Function is defined as:

$$\mu_i = \mathrm{e}^{-(x-m_i)^{\mathrm{T}}\Sigma_i^{\mathrm{T}}\Sigma_i(x-m_i)} \tag{1}$$

where x is the state of the vehicle, m_i the center (operating state) of the mode, and Σ_i is the positive semi-definite diagonal matrix whose elements represent the inverse of the deviations for each component of x for that mode.

To determine which mode is active, the MTM computes the Mode Membership Functions for all local modes. If $\mu_l(x(k)) \geq 0.5$ for the actual state, then local mode l will be active. Mode centers and deviations are defined so that $\mu_l(x(k)) \geq 0.5$ can be valid for only one l. That way the modes correspond to disjoint regions of the state space. If $\mu_l(x(k)) < 0.5$ for all l, then the transition corresponding to the two modes with the highest Mode Membership Function values will be active.

When a local mode is active, the corresponding local controller is used to compute the control output whereas when a transition is active, the corresponding active control model (ACM) is used to compute the control output. When a faulty mode is detected by the fault detection and identification (FDI) component, the system will transition to a fault tolerant control mode, where corresponding control reconfiguration tasks are executed.

6.2. Local controllers

The local controllers are of the discrete time tracking variety running at a fixed sample rate. The control law for these controllers is given by:

$$u(k) = K_i e(k) + u_{\mathrm{trim},i} \tag{2}$$

where k represents the discrete time, $u(k)$ is the actuator command vector, $e(k)$ the error between the desired state (set point) generated by the trajectory planning component ($x_{\mathrm{d}}(k)$) and the actual state of the vehicle obtained from onboard sensors ($x(k)$). The parameters for local controller i are the matrix gain K_i, and the trim value of the actuator command $u_{\mathrm{trim},i}$.

The state of the vehicle is given by:

$$x(k) = [x, y, z, \phi, \theta, \psi, u, v, w, p, q, r]^{\mathrm{T}}$$

where x: x-position (ft, measured northwards); y: y-position (ft, measured eastwards); z: z-position (ft, measured downwards); ϕ: roll angle (rad); θ: pitch angle (rad); ψ: yaw angle (rad); u: x-velocity (ft/s); v: y-velocity (ft/s); w: z-velocity (ft/s); p: roll rate (rad/s); q: pitch rate (rad/s); r: yaw rate (rad/s).

The actuator command vector is given by

$$u(k) = [\delta_{\mathrm{t}}, \delta_{\mathrm{coll}}, \delta_{\mathrm{lon}}, \delta_{\mathrm{lat}}, \delta_{\mathrm{tr}}]^{\mathrm{T}},$$

where δ_t: throttle; δ_{coll}: collective; δ_{lon}: longitudinal cyclic (moment actuator for pitch); δ_{lat}: lateral cyclic (moment actuator for roll); δ_{tr}: pedal (moment actuator for yaw).

A transformation is performed on $x(k)$ and $x_{\mathrm{d}}(k)$, before the control algorithms are applied, to make them independent of the actual heading of the vehicle. That is, if ψ_x is the actual value of the heading in $x(k)$, then the transformed values are obtained by

$$\begin{aligned} x(k) &\leftarrow T(x(k), \psi_x) \\ x_{\mathrm{d}}(k) &\leftarrow T(x_{\mathrm{d}}(k), \psi_x) \end{aligned} \tag{3}$$

where

$$T(x(k), \psi_x) = [x_{\psi_x}, y_{\psi_x}, z, \phi, \theta, \psi - \psi_x, u_{\psi_x}, v_{\psi_x}, w, p, q, r]^{\mathrm{T}}$$

with

$$\begin{bmatrix} x_{\psi_x} \\ y_{\psi_x} \end{bmatrix} = A(\psi_x)\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} u_{\Psi_x} \\ v_{\Psi_x} \end{bmatrix} = A(\psi_x)\begin{bmatrix} u \\ v \end{bmatrix}$$

$$A(\psi_x)\begin{bmatrix} \cos(\psi_x) & \sin(\psi_x) \\ -\sin(\psi_x) & \cos(\psi_x) \end{bmatrix}$$

After the transformation, the tracking error is given by

$$e(k) = x_{\mathrm{d}}(k) - x(k) \tag{4}$$

To improve the tracking performance of the local controllers, they are augmented with an integral part; therefore, the dynamics of the system is augmented with integrators for position, heading, and rotor angular velocity.

This is equivalent to designing the controllers for a system with state vector

$$x(k) = \left[\int x, \int y, \int z, \int \phi, \int \Omega, x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \Omega \right]^{\mathrm{T}}$$

The error in (4) is computed based on this augmented state vector.

The design procedure for the local controllers is as follows: once the operating state of a mode is decided, an approximate model of the vehicle is linearized about that state, and then discretized. A linear quadratic regulator is computed for the matrix gain K_i and the same design procedure is used for each mode. When an approximate model of the vehicle is not available, the linearized model could be obtained from a Fuzzy Neural Net (FNN) model trained with input–output data from the actual vehicle in the same way as with the active plant models to be discussed later.

6.3. Active control models

The active control models are in charge of the transitions between local modes. The function of an active control model is to blend the outputs of the local controllers corresponding to one transition in a smooth and stable way, that is, the blending of the local controllers should not deteriorate the overall performance of the closed loop system. Every ACM is linked to the local controllers corresponding to the transition, has access to their outputs, and also includes a Fuzzy Neural Net that generates the blending gains to compute the control output, as depicted in Fig. 13.

The FNN has the same structure as in (Rufus, 2001; Rufus et al., 2002), but its learning capabilities have been improved via a new recursive least squares training algorithm (Gutierrez, 2004). The input of the FNN is the actual state of the vehicle, $x(k)$, after the transformation given in (3).

Therefore, the output of the lth ACM module is determined from

$$\begin{aligned} &\mathrm{blendingGains} = \mathrm{FNN}_{\mathrm{ACM}_l}(x(k)) \\ &u(k) = \mathrm{blendingGains}(1)u_i(k) + \mathrm{blendingGains}(2)u_j(k) \end{aligned} \tag{5}$$

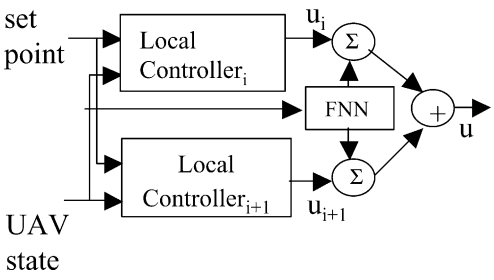

Fig. 13. Active control models structure.

where $\mathrm{FNN}_{\mathrm{ACM}_l}$ represents the function implemented by the FNN of the lth ACM, blendingGains is the output vector of that FNN, and $u_i(k)$ and $u_j(k)$ represent the control outputs of the local controllers corresponding to the lth ACM. When a transition is set up, the FNN of the corresponding ACM is trained off-line on the basis of an input–output data set generated automatically from a hypothetical transition trajectory from the center of the initial mode to the center of the target mode. The state is taken from this trajectory and the desired blending gains (desired outputs of the FNN) are computed based on the Mode Membership Functions generated by the mode transition manager. That is, given that the state of the vehicle is $x(k)$ at some point over this hypothetical trajectory, and μ_i and μ_j are the Mode Membership Functions for the modes involved in the transition from mode i to mode j, then the desired output for the FNN at that point is

$$\left[\frac{\mu_i(x(k))}{\mu_i(x(k)) + \mu_j(x(k))} \; \frac{\mu_j(x(k))}{\mu_i(x(k)) + \mu_j(x(k))} \right]^{\mathrm{T}}$$

At run time, the FNN of the ACM is adapted on-line by the control adaptation mechanism, as is described in the sequel.

Once the local modes are defined and the local controllers are designed for each local mode, the transitions are established via the ACM's in the mode transition control component and the corresponding active plant models, which are incorporated into the adaptation mechanism.

6.4. Adaptation mechanism component

The adaptation mechanism component calls the adaptation routines of the mode transition control and also includes the active plant models (one for each transition), which to serve as partial models of the plant in the transitions.

6.4.1. Active plant models

For each transition there is an ACM in the MTC component and the associated active plant model (APM) in the adaptation mechanism component. The purpose of the APM's is to serve as partial models of the plant in the transitions and provide the sensitivity matrices required to adapt the ACM's. Every APM includes a FNN that is trained to represent the dynamics of the vehicle in the transition region corresponding to that APM. Therefore, if the model of the vehicle is given by

$$x(k+1) = f(x(k), u(k)) \quad \text{with} \quad x(0) = x_0 \tag{6}$$

then, the FNN in the APM l is trained such that

$$\mathrm{FNN}_{\mathrm{APM}_l}(x(k), u(k)) \approx x(k+1) = f(x(k), u(k)) \tag{7}$$

given that transition l is active.

A recursive least squares training method minimizes the approximation error in (7), so this approximation is valid

when enough input/output data are available to train the FNN.

Near the actual operating point, defined by the pair $(x(k),u(k)) = (x_*,u_*)$, a linearized model of the vehicle is obtained from the FNN, that is

$$\Phi = \left.\frac{\partial f(x(k),u(k))}{\partial x(k)}\right|_{x_*,u_*} \approx \left.\frac{\partial \mathrm{FNN}_{\mathrm{APM}_l}(x(k),u(k))}{\partial x(k)}\right|_{x_*,u_*} \quad (8a)$$

$$\Gamma = \left.\frac{\partial f(x(k),u(k))}{\partial u(k)}\right|_{x_*,u_*} \approx \left.\frac{\partial \mathrm{FNN}_{\mathrm{APM}_l}(x(k),u(k))}{\partial u(k)}\right|_{x_*,u_*} \quad (8b)$$

so

$$\begin{aligned} x(k+1) &= f(x(k),u(k)) \\ &\approx \Phi(x(k)-x_*) + \Gamma(u(k)-u_*) + x_* \end{aligned} \quad (9)$$

Sensitivity matrices computed from (8a), and (8b) are used in the control adaptation mechanism to adapt the ACM's as is described below.

6.4.2. Plant adaptation mechanism

The plant adaptation mechanism is used to train the APM's. When the vehicle is in a transition, the input/output information from its sensors is used by the plant adaptation mechanism to train this model by calling the recursive least squares training routine from the FNN. The plant adaptation mechanism can be disabled at any time to free system resources, if required. In that case, the last value of the APM is used by the control adaptation mechanism to compute the sensitivity matrices.

6.4.3. Control adaptation mechanism

The control adaptation mechanism provides the adaptation function to the ACM's. When an ACM is active and the control adaptation mechanism is enabled, an optimization routine is used to find the optimal control value at each time step; the optimal blending gains that minimize the error between the optimal control and the control produced by the ACM are also computed. These optimal blending gains constitute the desired outputs for the recursive least squares training algorithm in the FNN, corresponding to that ACM, which is in turn called by the control adaptation mechanism.

The optimization routine used to compute the optimal control value uses a finite horizon optimal control methodology; the latter is based on the linearized model of the vehicle, which is obtained in turn from the sensitivity matrices generated from the corresponding APM, as given by (8a), (8b) and (9). The objective of this optimal control problem is to minimize the following performance index

$$J = \frac{1}{2}\sum_{i=k}^{k+N} e^{\mathrm{T}}(i)Qe(i) + \Delta u^{\mathrm{T}}(i)R\,\Delta u(i) \quad (10)$$

with $Q \geq 0$, $R > 0$

$$\text{subject to } \Delta x(i+1) = \Phi\,\Delta x(i) + \Gamma\,\Delta u(i) \quad (11)$$

for $i = k, k+1, \ldots, k+N$ with $\Delta x(k) = \Delta x_k$, where

$e(i) = x_d(i) - x(i)$

$\Delta x(i) = x(i) - x_*$

$\Delta u(i) = u^*(i) - u_*$

Application of the optimization algorithm gives the value of $\Delta u(k)$ which, in turn, is needed to compute $u^*(k)$ from

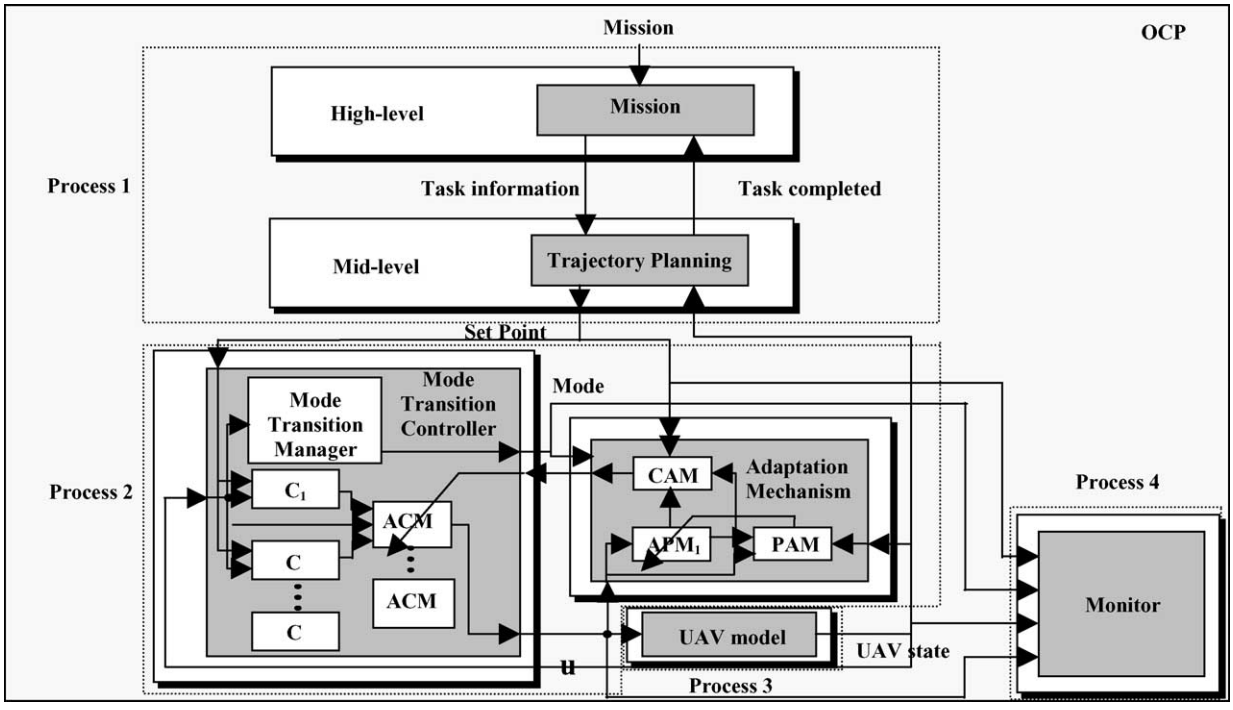

Fig. 14. Adaptive mode transition control implementation on the OCP.

$u^*(k) = u_* + \Delta u(k)$. This is the optimal control value used to compute the desired blending gains for the active control model.

The approach constraints the blending gains so the ACM produces a convex combination of the outputs of the local controllers and guarantees smooth transitions. That is, given the outputs of the local controllers corresponding to the ACM, $u_i(k)$ and $u_j(k)$, the objective is to minimize the magnitude of the error

$$\left\| u^*(k) - \text{desiredGains}(1)u_i(k) + \text{desiredGains}(2)u_j(k) \right\|_2^2$$

subject to

$$0 \le \text{desiredGains}(i) \le 1 \quad \text{for} \quad i = 1, 2$$
$$\text{desiredGains}(1) + \text{desiredGains}(2) = 1$$

A simple algorithm achieves this objective:

$$\alpha = \text{sat}\left(\frac{\Delta u(u^*(k) - u_i(k))}{\Delta u\, \Delta u}\right)$$

$$\text{desiredGains}(1) = 1 - \alpha$$
$$\text{desiredGains}(2) = \alpha$$

where $\Delta u = u_j(k) - u_i(k)$

$$\text{sat}(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

These desired gains become the desired outputs for the recursive least squares algorithm, which trains the FNN of the ACM.

6.5. Implementation and flight test results

The architecture has been implemented using the OCP. Fig. 14 shows a software-in-the-loop simulation environment used to implement the architecture. Hardware-in-the-loop simulations and flight tests have been performed to

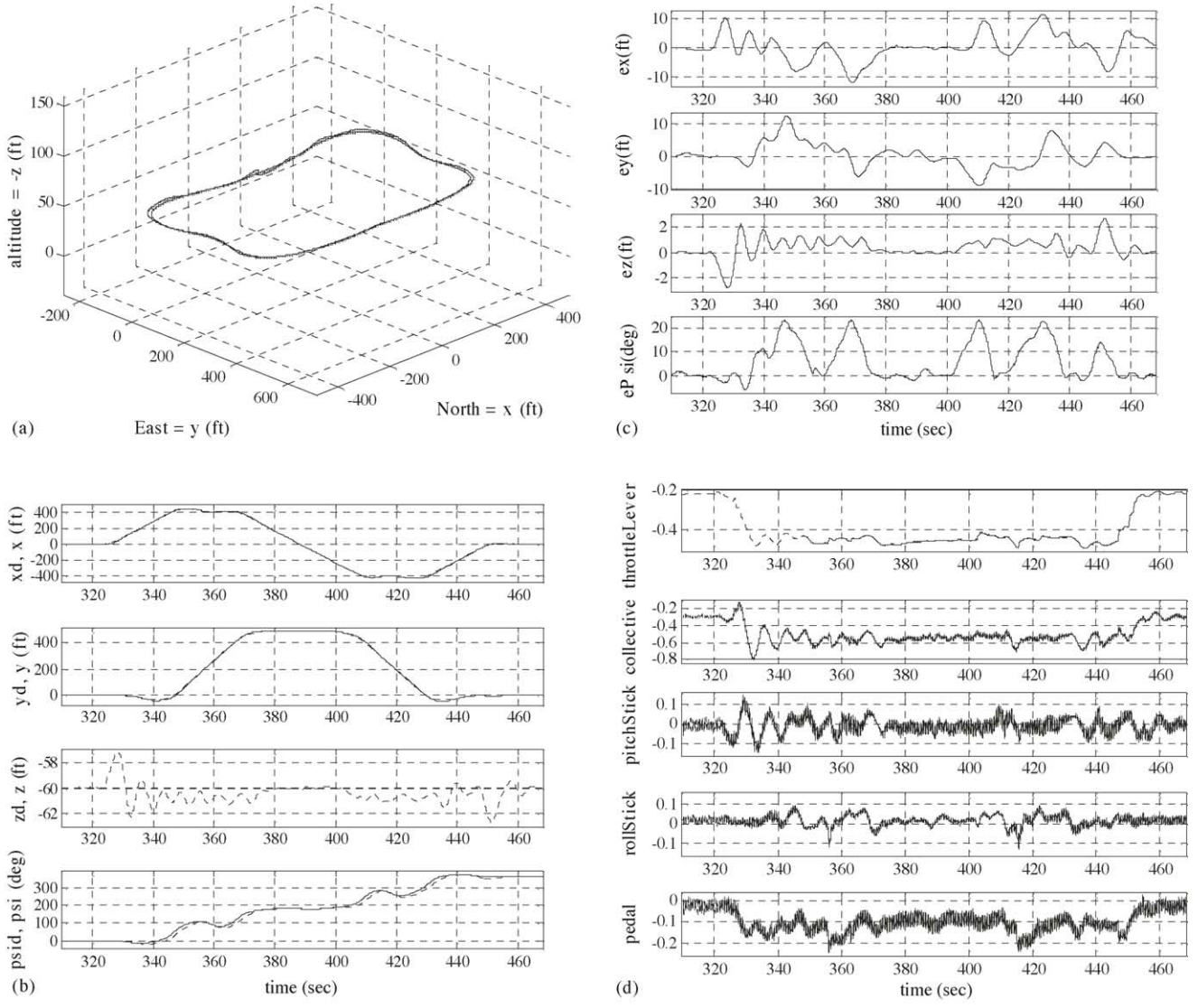

Fig. 15. AMTC algorithms flight test results: (a) desired and actual 3D trajectory; (b) desired and actual position and heading; (c) position and heading errors; (d) actuator commands.

validate the control algorithms. Fig. 15 shows the flight test results of a rectangular flight.

7. Fault tolerant control

UAVs are often subjected to failure modes that may lead to a catastrophic event resulting in loss of the vehicles. It is desired, therefore, to develop and implement technologies that will detect and identify in a timely manner to onboard failure modes and reconfigure the available control authority so that the vehicle maintains an acceptable level of performance for the duration of the emergency (Clements, 2003; Clements, Heck, & Vachtsevanos, 2001). The fault tolerant control architecture implemented on the GTMax is designed to accommodate multiple fault modes without degrading the performance of the nominal system. The architecture improves reliability by integrating fault detection and identification and reconfigurable flight control (RFC).

Implementation of the architecture utilizes a three-tier hierarchical control scheme implemented in the OCP (Fig. 10). The architecture is a variation of the scheme developed by Clements (2003). Each level of the hierarchy adds autonomy to the vehicle. FDI takes place at the highest level of the hierarchy and directs actions at each subordinate tier. After the FDI module issues a fault declaration, the fault tolerant control module issues reconfiguration commands to the controllers at the lowest tier of the hierarchy. Reconfigurable flight controllers reside with the baseline flight controller at the lowest level of the hierarchy. The low-level controllers generate the control inputs to achieve the vehicle's desired flight path. In the event of a malfunction, reconfigurable flight controllers enable the vehicle to recover some degree of the performance from the impaired system.

The architecture implemented on the GTMax was designed to combat faults in the flight control actuators. Malfunctions in the flight control hardware were selected for this study because they challenge both components of the fault tolerant control architecture, FDI and RFC. Specifically, a malfunction in the main rotor collective actuator was examined, but the architecture is readily expandable to accommodate additional faults. The design identifies the occurrence of a fault from a finite set of pre-determined faults. It then applies the appropriate reconfiguration to stabilize the vehicle. The fault tolerant architecture assumes the following actuator model:

$$\delta = \max(\min(k\delta_{\text{com}} + b, s_{\max}), s_{\min})$$

where faults can affect the actuator gain k: $0 \leq k \leq 1$, bias b, or saturation levels, $s_{\min}$ and $s_{\max}$. Faults, such as floating actuators, where the parameters vary constantly following the occurrence of the fault are not considered. To accommodate these cases, additional hardware on the vehicle could be employed to immobilize the actuator creating a stuck actuator condition. Flight test results demonstrate that the fault tolerant architecture can accommodate stuck actuator malfunctions with $k = 0$.

7.1. Fault detection and identification

The success of the prescribed architecture depends entirely on the success of the fault detection and identification algorithm. FDI must occur quickly, that is within a few seconds, or the degraded system will readily depart from the flight envelope of the reconfigurable flight controller. A small number of false positives are acceptable. False negatives and mis-identifications typically result in loss of the aircraft. Furthermore, the FDI must be robust in design so that neither adverse environmental conditions nor aggressive flight trajectories degrade their performance. The implemented FDI routine is state-dependent; that is it detects faults based solely on the vehicle's flight dynamics. No additional sensors were installed to aid fault detection.

The FDI routine detects a loss of collective pitch control in the main rotor system of the GTMax. A fault of this kind, if not detected and compensated for, can quickly precipitate a catastrophic failure of the vehicle. However, a false positive activates the reconfigurable flight controller, which results in reduced flight capabilities of the vehicle. Hence, the goal is to design a FDI methodology with a good balance of sensitivity and reliability.

At the heart of the FDI technique discussed below is a three-layer feed-forward neural network. This neural network is trained off-line by back propagation using both simulation and flight data to finalize the weights. The network is then used for real time fault detection. In order to make the algorithm more effective, the training set needs to be extensive with respect to flight profiles and environmental conditions. Additionally, the input vector must be chosen so that the state and control vectors included have a significant correlation with the fault mode under investigation. For the case in point, the signals of interest were the magnitude of the position, velocity and acceleration errors in the vertical axis, the commanded collective pitch and the main rotor speed.

To further refine the method, the input signals were passed through an averaging filter to eliminate high frequency noise and to enhance the signature trends produced by the collective pitch fault. This together with adequate training of the neural network ensures satisfactory sensitivity of the algorithm to collective faults. To improve the rejection of false positives and increase confidence in the method, a thresholding condition is imposed on the network output. The threshold limit is a parameter that can be tuned to obtain a good balance between sensitivity and reliability.

7.2. Reconfigurable flight controller

The reconfigurable flight controller developed exploits an unconventional control strategy to expand the controllability

of the fault-impaired system. The control vector for a typical helicopter includes four inputs: collective, δ_{coll}, lateral cyclic, δ_{lat}, longitudinal cyclic, δ_{lon}, and tail rotor pitch, δ_{tr}. In such a system, a throttle control loop or an engine governor manipulates the throttle, δ_t, to maintain the speed of the main rotor, Ω, at a constant value, Ω_{com}. Constructing a throttle outer loop that controls Ω_{com} augments the controllability of the aircraft. Adequate performance from the baseline throttle inner loop allows use of this strategy for reconfigurable flight control.

Helicopter vertical thrust is a strong function of both δ_{coll} and Ω. In the nominal state, vertical thrust is controlled by δ_{coll} with Ω held constant. The converse of this control strategy is feasible with a loss in response time because variation of Ω requires adding or subtracting rotational energy from the rotor system. The case where δ_{coll} is held stationary represents the most severe class of malfunction. Partially degraded response from δ_{coll} provides an improvement over the fully degraded case.

The baseline throttle inner loop controller achieves approximate adherence to the single pole linear system:

$$\dot{\Omega} = \frac{1}{\tau}(\Omega_{com} - \Omega)$$

The control Ω_{com} to achieve a desired rotor angular rate, Ω_{des} is thus:

$$\Omega_{com} = \tau\dot{\Omega}_{des} + \Omega_{des}$$

A feedback linearization type controller is used to control the vertical thrust of the degraded system. Assuming the plant has affine dynamics:

$$\dot{w} = f + g\Omega$$

where $\dot{w}$ is the translational velocity of the vehicle in the body z-direction. The following control is applied to the system:

$$\Omega_{des} = \frac{1}{g'}[\dot{w}_{com} - f' - K_d(w - w_{com}) - K_p(z - z_{com})]$$

where $\dot{w}_{com}$, w_{com}, and z_{com} indicate the desired vertical dynamics of the vehicle. A second order filter is normally applied to these dynamics prior to inclusion in the control law above. Changing parameters in this filter provides a means to smooth flight paths after the occurrence of a fault. f' and g' are estimates of the actual f and g such that

$$\dot{w}' = f' + g'\Omega$$

where f' is estimated by a linear combination of the state and control vectors plus the output of an adaptive neural network. The input layer of the network consists of the vehicle state and control vectors and the linear estimate. Back propagation updates the single hidden layer network to minimize the model error, ε. g' is approximated simply as a constant. Scheduling g' with horizontal velocity has also proved to be beneficial in simulation.

$$\varepsilon = \dot{w} - \dot{w}'$$

Combining equations for Ω_{com} and Ω_{des}, and letting $e = z - z_{com}$, one obtains the following:

$$\Omega_{com} = \frac{1}{g'}[\tau\dddot{w}_{com} + \ddot{w}_{com} - \tau\ddot{f}' - \dot{f}' - \tau K_d\dddot{e}. \\ -(\tau K_p + K_d)\ddot{e} - K_p\dot{e}]$$

The error dynamics of the system reduce to:

$$\dddot{e} = -\left(K_d + \frac{1}{\tau}\right)\ddot{e} - \left(\frac{K_d}{\tau} + K_p\right)\dot{e} - \frac{K_p}{\tau}e + \frac{\varepsilon}{\tau} + \dot{\varepsilon}$$

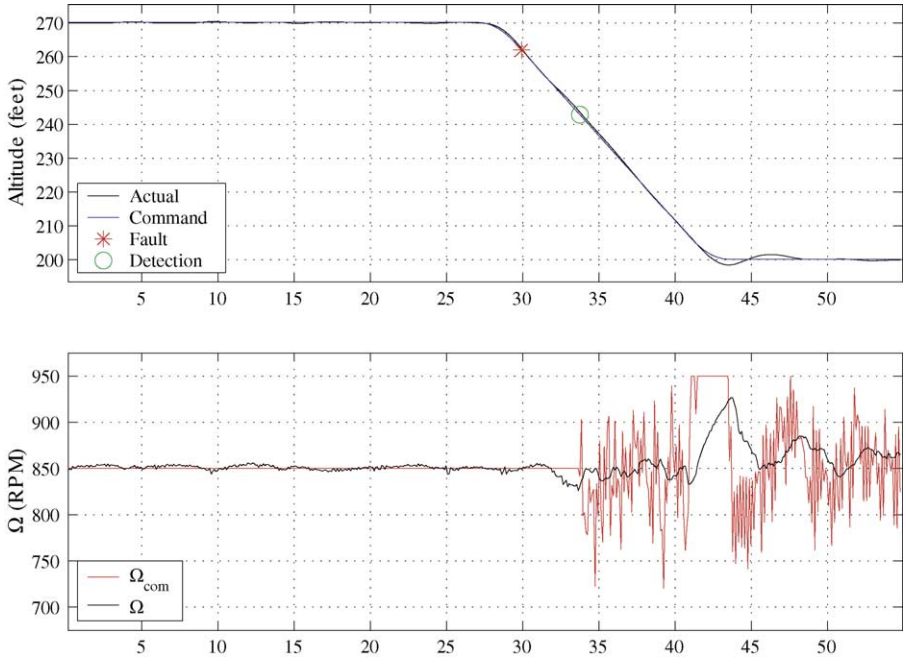

Fig. 16. Fault tolerant control flight test result.

Assuming convergence of the network, ε and $\dot{\varepsilon}$ are negligible. The characteristic equation takes the following form:

$$\left(s+\frac{1}{\tau}\right)(s^2+2\zeta\omega_n+\omega_n^2)=0$$

K_p and K_d are set as follows to dictate the performance of the second order system:

$$K_\mathrm{p}=\omega_n^2$$
$$K_\mathrm{d}=2\zeta w_n$$

This method of adaptive neural network control is simple in implementation, but it lacks certain stability properties that are assured by more advanced adaptive neural network flight controllers (Johnson & Kannan, 2002). The proposed controller depends on the assumption that the neural network converges using standard back propagation. In the event the neural network fails to converge, a PID controller is activated. Through several flight tests, the control strategy has performed well and the neural network has never diverged.

Flight test results validate the effectiveness of the approach. The flight demonstration was initiated with the UAV in its baseline configuration. The aircraft was commanded to execute a 70-ft descent from a stationary hover. During the descent, the stuck collective fault was applied binding the collective in a typical descent position that was determined from flight data on the day of the flight test. The state-dependent neural network FDI routine detected the fault and activated system restructuring. Referencing Fig. 16, the descent is initiated at 28 s; the fault is applied at 30 s, and the fault is detected prior to 34 s. Without reconfiguration, the vehicle would not have been able to arrest its descent.

8. Conclusion

Unmanned aerial vehicles present major challenges to the designer and the end user. They require new and novel technologies to be developed, tested and implemented if such vehicles will perform actual missions reliably and robustly. Autonomy stands out as the key requirement with enabling technologies to allow such vehicles to operate safely in unstructured environments within their flight envelope, to accommodate subsystem/component failure modes without major performance degradation or loss of vehicle and to perform extreme maneuvers without violating stability limits. An integrated/hierarchical approach to vehicle instrumentation, computing, modeling and control seems to provide possible solutions. The UAV community is accomplishing major milestones towards this goal but key R&D concerns remain to be addressed. More recently, researchers have been concerned with multiple and heterogeneous UAVs flying in formation in order to take advantage of their complementary capabilities (Vachtsevanos, Tang, & Reimann, 2004). The UAV swarm problem opens now avenues of research where the intelligent control community can contribute significantly in terms of smart coordination/cooperation technologies.

Acknowledgements

The authors would like to gratefully acknowledge the support received from Dr. John. S. Bay, DARPA, and Mr. William Koenig, AFRL. In addition, the authors would like to thank their Co-P.I.'s on this research project: Dr. Daniel Schrage, Dr. J.V.R. Prasad, Dr. Eric Johnson, School of Aerospace Engineering, and Drs. Linda Wills and Bonnie Heck, School of Electrical and Computer Engineering, Georgia Tech; and Mr. Bryan Doerr, Mr. Brian Mendel, and Mr. James L. Paunicka, Boeing Phantom Works.

References

Al-Hasan, S., & Vachtsevanos, G. (2002). Intelligent route planning for fast autonomous vehicles operating in a large natural terrain. *Journal of Robotics and Autonomous Systems, 40*, 1–24.

Cardei, M., Cardei, I., Jha, R., & Pavan, A. (2000). Hierarchical feedback adaptation for real time sensor-based distributed applications. In *Proceedings of the Third IEEE International Symposium on Object-oriented Real-time Distributed Computer (ISORC)* (pp. 181–188).

Clements, N.S. (2003).*Fault tolerant control of complex dynamical systems fault tolerant control of complex dynamical systems*. Ph.D. Thesis, Georgia Tech.

Clements, N. S., Heck, B., & Vachtsevanos, G. (2001). Hierarchical fault tolerant control using component based models. In *Proceedings of the 2001 European Control Conference*.

DARPA. (2004). Software enabled control. Department of Advance Research Projects, Information Expoitation Office. http://dtsn.darpa.mil/ixo/programs.asp.

DoD. (2002). UAV Roadmap 2002–2027. Office of the Secretary of Defense (Acquisition, Technology, & Logistics), Air Warfare. http://www.acq.osd.mil/usd/uav_roadmap.pdf.

Guler, M., Clements, S., Wills, L., Heck, B., & Vachtsevanos, G. (2003). Transition management for reconfigurable hybrid systems. *IEEE Control Systems Magazine, 23*(February (1)), 36–49.

Gutierrez, L. (2004). *Adaptive mode transition control architecture with an application to unmanned aerial vehicles*. Ph.D. Thesis, Georgia Tech.

Gutierrez, L., Vachtsevanos, G., & Heck, B. (2003a). An approach to the adaptive mode transition control of unmanned aerial vehicles. In *Proceedings of 2003 American Control Conference*.

Gutierrez, L., Vachtsevanos, G., & Heck, B. (2003b). A hierarchical architecture for the adaptive mode transition control of unmanned aerial vehicles. In *Proceedings of AIAA Guidance, Navigation and Control Conference*.

Johnson, E., & Kannan, S. (2002). Adaptive flight control for an autonomous unmanned helicopter. *AIAA Guidance, Navigation and Control Conference*.

Levine, D., Gill, C., & Schmidt, D. (1998b). Dynamic scheduling strategies for avionics mission computing. In *Proceedings of 17th Digital Avionics Systematic Conference, Vol. 1* (pp. C151–C158).

Levine, D., Mungee, S., & Schmidt, D. (1998a). The design and performance of real-time object request brokers. *Computer Communication, 21*(April), 294–324.

Rufus, F., (2001). *Intelligent approaches to mode transition control*. Ph.D. thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA.

Rufus, F., Heck, B., & Vachtsevanos, G. (2000b). Software-enabled adaptive mode transition control for autonomous unmanned vehicles. In *Proceedings 19th Digital Avionics Systems Conference, Vol. 1* (pp. 1E.1-1–1E.1-8).

Rufus, F., Vachtsevanos, G., & Heck, B. (2000a). Adaptive mode transition control of nonlinear systems using fuzzy neural networks. *Eighth IEEE Mediterranean Conference on Control and Automation*.

Rufus, F., Vachtsevanos, G., & Heck, B. (2002). Real-time adaptation of mode transition controllers. *Journal of Guidance, Control, and Dynamics, 25*(1), 167–175.

Schmidt, D., & Kuhns, F. (2000). An overview of the real-time CORBA specification. *IEEE Computer, 33*(June), 56–63.

Schmidt, D., Levine, D., & Harrison, T. (1997). The design and performance of a real-time CORBA event service. In *Proceedings of the OOPSLA'97* (pp. 184–200).

Schrage, D. P., & Vachtsevanos, G. (1999). Software enabled control for intelligent UAVs. *1999 IEEE International Conference on Control Applications*.

Schrage, D. P., et al. (1997). *Autonomous Scout Rotorcraft Testbed (ASRT) Final Report*, Georgia Tech, July 24.

Vachtsevanos, G., Kim, W., Al-Hasan, S., Rufus, F., & Simon, M. (1997a). Autonomous vehicles: From flight control to mission planning using fuzzy logic techniques. *International Conference on Digital Signal Processing, DSP'97* (pp. 977–981).

Vachtsevanos, G., Kim, W., Al-Hasan, S., Rufus, F., Simon, M., Schrage, D., et al. (1997b). Mission planning and flight control: Meeting the challenge with intelligent techniques. *Journal of Advanced Computational Intelligence, 1*(1), 62–70.

Vachtsevanos, G., Tang, L., & Reimann, J. (2004). An intelligent approach to coordinated control of multiple unmanned aerial vehicles. *American Helicopter Society 60th Annual Forum.*

Wills, L., Kannan, S., Heck, B., Vachtsevanos, G., Restrepo, C., Sander, S., et al. (2000b). An open software infrastructure for reconfigurable control systems. In *Proceedings of the 2000 American Control Conference, Vol. 4* (pp. 2799–2803).

Wills, L., Sander, S., Kannan, S., Kahn, A., Prasad, J. V. R., & Schrage, D. (2000a). An open control platform for reconfigurable, distributed, hierarchical control systems. In *Proceedings 19th Digital Avionics Systems Conferences, Vol. 1* (pp. 4D2/1–4D2/8).

Wills, L., Kannan, S., Sander, S., Guler, M., Heck, B., Prasad, J. V. R., et al. (2001). An open platform for reconfigurable control. *IEEE Control Systems Magazine, 21*(3), 49–64.

Yavrucuk, I., Unnikrishnan, S., & Prasad, J. V. R. (2003). Envelope protection in autonomous unmanned aerial vehicles. *59th AHS Annual Forum, May.*

George J. Vachtsevanos attended the City College of New York and received his BEE degree in 1962. He received an MEE degree from New York University and his PhD degree in electrical engineering from the City University of New York in 1970. His research focused on adaptive control systems. After graduate school, he taught within the City University System of New York from 1970 through 1975; from 1975 through 1977 he served as an associate professor of electrical engineering at Manhattan College. In 1977 he was elected to a chair professorship in electrical engineering at the Democritus University of Thrace, Greece, where he served as the pro-rector during the 1978–1979 academic year. He joined the Georgia Institute of Technology as a visiting professor in 1982–1983 and as a professor in 1984. He is also serving as an adjunct faculty member in the School of Textile and Fiber Engineering. He was the 3M McKnight distinguished visiting professor at the University of Minnesota, Duluth during the 1994 Spring Semester. Since joining the faculty at Georgia Tech, Dr. Vachtsevanos has been teaching courses and conducting research on intelligent systems, robotics and automation of industrial processes and diagnostics/prognostics of large-scale complex systems.

Liang Tang received a BE in industry automation (1994), a BSc in applied mathematics (1994) and a PhD in control theory and control engineering (1999) from Shanghai Jiao Tong University. He worked for Ericsson R&D companies in China and later in Japan for 4 years before he joined Intelligent Control Systems Laboratory as a postdoctoral fellow. Dr. Tang is currently employed by Impact Technologies, Rochester, New York. His current research interests include intelligent control, cooperative control, vision-based control of UAV/MAV systems, wireless communication and control systems.

Graham R. Drozeski is an electrical and computer engineering graduate student working in the Intelligent Control Systems Laboratory at the Georgia Institute of Technology, Atlanta, GA. His research interests include fault tolerance, reconfigurable flight control, and reconfigurable path planning. Mr. Drozeski holds a MS degree in aerospace engineering from the Georgia Institute of Technology, 2003 and a BS in electrical engineering from the University of Notre Dame, 1993. He served as helicopter pilot in the U.S. Army for 9 years prior to commencing his graduate studies.

Luis Gutiérrez was born in Medellín, Colombia. He received the electronics engineer degree from Universidad Pontificia Bolivariana (Medellín, Colombia) in 1989 and the MSc in electrical engineering from The University of Texas at Arlington (Arlington, Texas) in 1996. He developed his Master Thesis at the Automation and Robotics Research Institute under the advising of Dr. Frank L. Lewis. In 1989, Dr. Gutiérrez joined the faculty at the School of Electrical and Electronics Engineering, Universidad Pontificia Bolivariana. Currently, he is researcher of the A + D research group. Dr. Gutierrez received a PhD degree in electrical and computer engineering at Georgia Institute of Technology (Atlanta, Georgia) under the advising of Dr. George Vachtsevanos in 2004. Dr. Gutiérrez was awarded a Fulbright-LASPAU scholarship during his master's studies, and a Fulbright-LASPAU-Colciencias scholarship for his PhD studies. He is author of the book "*Sistemas y Señales*" and he has coauthored several journal and conference papers. His research interests are control theory, intelligent control systems, autonomous vehicles, and signal processing.