A HIERARCHICAL/INTELLIGENT CONTROL ARCHITECTURE FOR UNMANNED AERIAL VEHICLES

Luis B. Gutiérrez, George Vachtsevanos, and Bonnie Heck,
School of Electrical and Computer Engineering, Georgia Institute of Technology,
Atlanta, Georgia 30332-0250

Abstract

In this paper, a hierarchical/intelligent control architecture for an unmanned aerial vehicle (UAV) is proposed. The architecture consists of three levels: the highest level is occupied by mission planning routines. At this level, information about the way points the vehicle must follow is available and logic-based routines decide upon mission tasks while maintaining physical constraints and generate the task queue. The mid-level controller coordinates the task execution while a trajectory planning component receives the task information from the high-level module and provides set points for low-level stabilizing controllers whose function is to maintain the vehicle in a stable state and to follow accurately the commanded trajectory. An adaptive mode transitioning control algorithm resides also at the lowest level of the hierarchy consisting of two components: a mode transitioning controller and the accompanying adaptation mechanism. The adaptation routine may be turned on only when needed. The transitioning algorithm operates in real-time while adapting on-line to disturbances and other external inputs. This intelligent/hierarchical architecture is being implemented using a novel software infrastructure called Open Control Platform, which facilitates interoperability, plug-and-play and other functionalities. Simulation results illustrate the robustness and effectiveness of the proposed scheme. An actual flight demonstration is planned for the near future as part of a DARPA sponsored research program.

Introduction

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. These challenges appear also in other complex system applications, so new software enabled control technologies are being developed to address them [1]. In this paper, an intelligent/hierarchical control architecture for unmanned aerial vehicles (UAV) is proposed. 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 [2,3,4,5] where the adaptive mode transition control scheme was first introduced. This paper suggests a new approach to the adaptive mode transition control problem and introduces a hierarchical architecture to implement it. In this new approach, desired transition models are replaced by the middle level trajectory planning and the high level mission planning components. The architecture is flexible enough enable the future integration of additional intelligent attributes at the high level. A new adaptive mode transition control scheme and its associated algorithms are discussed. The algorithms have been implemented and tested using the Open Control Platform (OCP) - a new open software infrastructure especially developed for the implementation of complex reconfigurable control systems such as UAV's [6,7,8].

The paper is structured as follows: the overall architecture is presented first, and then its components are described; a detailed description of the new approach to the adaptive mode transition control is given; and finally, we comment on the OCP implementation, and present simulation results.

Overall Architecture For Control Of Unmanned Aerial Vehicles

The proposed architecture for the control of UAV's consists of a hierarchy of three levels (Figure 1). At the highest level, a mission planning component stores information about the overall

0-7803-7367-7/02/$17.00 © 2002 IEEE

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. At the lowest level, an adaptive mode transition controller coordinates the execution of the local controllers or the active control models, which stabilize the vehicle and minimize the errors between the set points generated by the middle level and the actual state of the vehicle. A more detailed description of each level as applied to the case of a rotary wing UAV is given below.

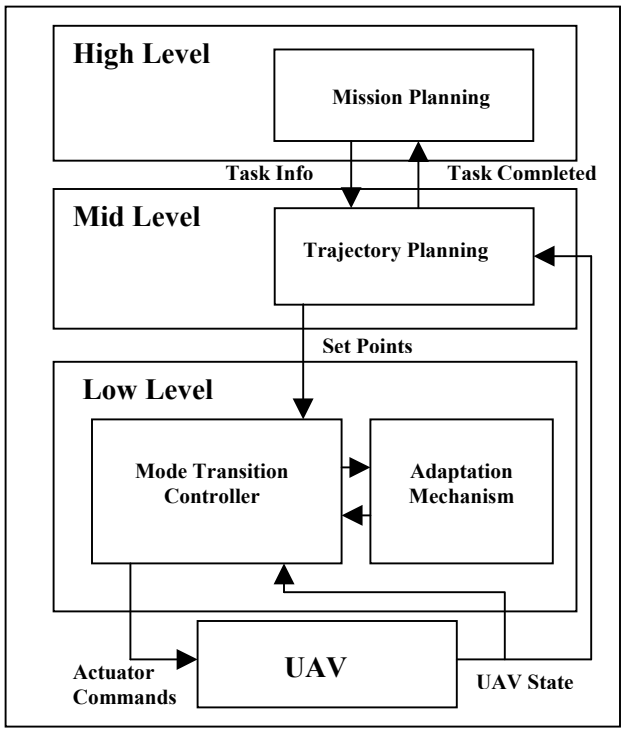

Figure 1. Hierarchical Control Architecture

High Level: Mission Planning

The mission planning component translates a high level representation of the mission into a low-level task queue and coordinates the execution of low-level tasks with the trajectory planning component at the middle level. The mission can be established as a sequence of actions to be executed, for instance: fly to a way point and hover there, fly to a way point at certain speed, keep the same velocity and heading for a certain period of time, etc. Cinematic constraints like maximum speed and acceleration are specified and can be changed for each section of the mission.

Every action is specified through a high level command given to the mission planning module. When a new action is suggested, the sequence of tasks that must be performed are generated and added to the tail end of the task queue. Each task represents a maneuver that takes the vehicle from the actual state to a target state and includes the following information:

- Time to complete the task
- Target position
- Target direction of the flight path for this task
- Target heading angle
- Heading mode: specifies whether the value of the heading is absolute or relative to the direction of the flight path for coordinated flight
- Target speed
- Maximum acceleration

A mission may be completely specified before it is executed but may also be modified, or re-planned or expanded at run time. This feature enables the modification or extension of the mission at run time. Re-planning is particularly important for the future incorporation of obstacle and collision avoidance algorithms.

At run time, the mission planning component coordinates the execution of the low-level tasks with the trajectory planning component at the middle level in the following way: first, the mission planning component takes the task at the head of the task queue, removes it from the queue and sends the task information to the trajectory planning component; then, the trajectory planning component executes the task and, when completed, it sends a signal back to the mission planning component indicating that the last task has been completed; finally, when the mission planning component receives the signal, places the task at the head of the task queue and the cycle is repeated until no tasks remain in the task queue.

Middle Level: Trajectory Planning

The trajectory planning component generates the set points required for the low-level controllers to complete the last task received from the mission planning module. When the trajectory planning component receives the next task information, it computes a 3D spline to generate a continuous path linking the actual position with the target position. At each sample time, the actual speed is evaluated based on the initial speed for the task, the final speed for the task, and the maximum acceleration available according to the curvature of the path at that time. Position and velocity over the path are computed next using the spline representation. A similar spline is used to represent the heading of the vehicle while the heading is determined in two ways according to the heading mode defined for the task: either directly from the heading spline if the heading mode is set to the absolute heading, or from a combination of the heading spline and the heading computed from the direction of the path, if the heading mode is set to the coordinated heading. Also the heading rate is computed in a consistent manner.

After generating the set points corresponding to the actual sample time, the condition for completion of the task is checked and a comparison is made with the actual state of the vehicle to determine if the task was completed successfully or not. A signal is sent to the high level module indicating the termination status of the task so that the next one can be initiated. For instance, when the mission planning component at the high level receives a signal of successful termination of the task, it retrieves the next task information from the task queue and sends it to the trajectory planning component at the middle level, so the trajectory generation continues smoothly. When the trajectory planning component completes a task and does not receive a new task to perform from the mission planning component, it generates set points consistent with the last set point, i.e. it maintains the same speed, path direction and heading, and computes the positions accordingly.

Low-Level: Adaptive Mode Transition Control

The purpose of the low level controllers is to stabilize the vehicle and force it to follow accurately the commanded trajectory generated by the middle level. In this architecture, a new approach to the adaptive mode transition control is introduced. A detailed description of the approach follows in the next section.

A New Approach To The Adaptive Mode Transition Control

The adaptive mode transition control consists of the mode transition control component and the adaptation mechanism component (Figure 2). The following description refers to the case of a rotary wing UAV.

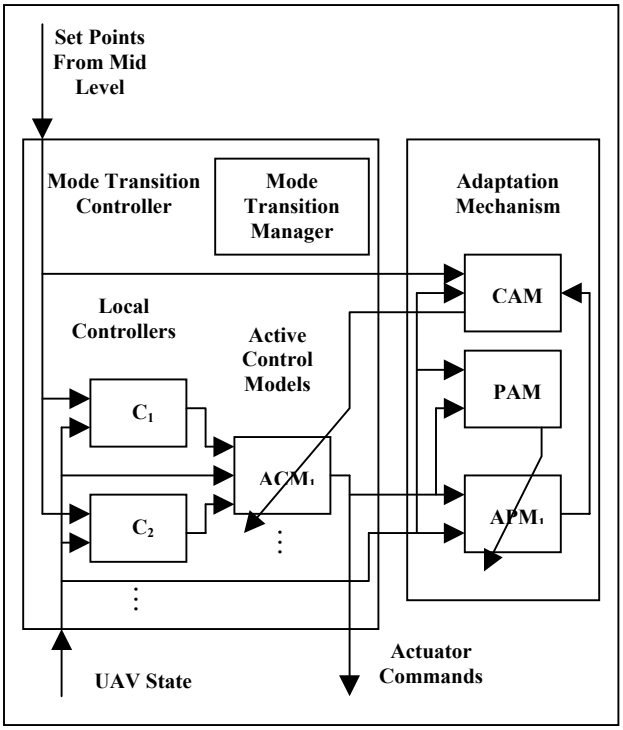

Figure 2. Adaptive Mode Transition Control

Mode Transition Control Component

The mode transition control component consists of several subcomponents: the local controllers (one for each local mode), the active control models (one for each transition), and the mode transition manager. The mode transition manager decides which controller to use at a given time (a local controller or an active control model) based on the actual state of the UAV. The mode transition control by itself does not perform any adaptation.

Local Controllers

In this new approach, 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:

$$u(k) = K_i e(k) + u_{trim,i} \qquad (1)$$

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

The state of the vehicle is given by

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

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/sec)

v : y-velocity (ft/sec)

w : z-velocity (ft/sec)

p : roll rate (rad/sec)

q : pitch rate (rad/sec)

r : yaw rate (rad/sec)

A transformation is performed on $x(k)$ and $x_d(k)$, 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

$$\begin{aligned} x(k) &\leftarrow T(x(k), \psi_x) \\ x_d(k) &\leftarrow T(x_d(k), \psi_x) \end{aligned} \qquad (2)$$

where

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

with

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

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

After the transformation, the tracking error is given by

$$e(k) = x_d(k) - x(k) \qquad (3)$$

The actuator command vector is given by

$$u(k) = [throttleLever, pitchStick, rollStick, pedal]^T$$

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 model trained with input-output data from the actual vehicle in the same way as with the active plant models to be discussed later.

Mode Transition Manager

The mode transition manager (MTM) coordinates the transitions in this new approach. Unlike [2,3,4,5] where the transitions were pre-scheduled and a Mode Selector module coordinated the transitions, the MTM coordinates the transitions automatically in the new technique 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

$$\mu_i = e^{-(x-m_i)^T \Sigma_i^T \Sigma_i (x-m_i)} \qquad (4)$$

where *x* is the state of the vehicle, m_i is the center (operating state) of the mode, and Σ_i is a 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 ACM is used to compute the control output.

Active Control Models

The active control models are in charge of the transitions between local modes. The function of an active control model (ACM) 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 (FNN) that generates the blending gains to compute the control output. The FNN has the same structure as in [2,3,4,5], but its learning capabilities have been improved via a new recursive least squares training algorithm. The input of the FNN is the actual state of the vehicle, $x(k)$, after the transformation given in (2). Therefore, the output of the lth ACM module is determined from

$$\begin{aligned} blendingGains &= FNN_{ACM_l}(x(k)) \\ u(k) &= blendingGains(1)u_i(k) + \\ &\quad blendingGains(2)u_j(k) \end{aligned} \qquad (5)$$

where FNN_{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. The new approach differs from the one presented in [2,3,4,5] in that it uses scalar blending gains, while in [2,3,4,5] different blending gains are used for each component of *u*(*k*).

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

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

Thus, the computation of an optimal trajectory for the given transition is avoided at this stage. This new approach assumes that the mode transition controller itself does not determine the trajectory for the given transition since the trajectory planning component specifies the trajectory at the middle level according to the tasks sent by the mission planning component.

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.

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. This concept is defined in next section.

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

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

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

$$FNN_{APM_l}(x(k),u(k)) \approx x(k+1) = f(x(k),u(k)) \qquad (7)$$

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

$$\Phi = \left.\frac{\partial f(x(k),u(k))}{\partial x(k)}\right|_{x_*,u_*} \approx \left.\frac{\partial FNN_{APM_l}(x(k),u(k))}{\partial x(k)}\right|_{x_*,u_*} \qquad (8a)$$

$$\Gamma = \left.\frac{\partial f(x(k),u(k))}{\partial u(k)}\right|_{x_*,u_*} \approx \left.\frac{\partial FNN_{APM_l}(x(k),u(k))}{\partial u(k)}\right|_{x_*,u_*} \qquad (8b)$$

so

$$x(k+1) = f(x(k),u(k)) \approx \Phi(x(k)-x_*) + \Gamma(u(k)-u_*) + x_* \qquad (9)$$

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

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.

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 (8) and (9). The objective of this optimal control problem is to minimize the following performance index

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

with $Q \geq 0, R > 0$

subject to

$$\Delta x(i+1) = \Phi\Delta x(i) + \Gamma\Delta u(i) \qquad (11)$$

for $i = k, k+1, ..., k+N$

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 $u^*(k) = u_* + \Delta u(k)$. 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

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

subject to

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

$$desiredGains(1) + desiredGains(2) = 1$$

A simple algorithm achieves this objective:

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

$$desiredGains(1) = 1 - \alpha$$
$$desiredGains(2) = \alpha$$

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

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

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

Implementation Based On The Open Control Platform (OCP)

The Architecture already described above has been implemented using an Open control Platform (OCP). The OCP is a software infrastructure developed by Boeing in collaboration with Georgia Tech to enable the implementation of advanced control algorithms for UAV's [6,7,8]. It allows for system reconfiguration, interoperability of different operating systems and platforms, plug and play connectivity, while it enables the implementation of sophisticated multirate hybrid systems. The OCP includes a Controls API which allows the user to generate easily the code required for the application at hand, and to customize it to include his own control algorithms. Georgia Tech is also developing a Hybrid Controls API that is being integrated into the OCP, which facilitates the implementation of certain common operations required for hybrid control systems [9,10].

The actual implementation of this hierarchical control architecture uses the controls API of the OCP. It is anticipated that the Hybrid Controls API will also facilitate the implementation of the algorithms of the mode transition control being hybrid in nature (transitions between different local controllers and active control models).

Figure 3 shows how the OCP is used to implement this hierarchical control architecture in a simulation mode. It will be integrated into a software-in-the-loop simulation of the UAV, then a hardware-in-the-loop simulation, and finally a flight test in the near future.

For this implementation, the components are distributed among four processes: one for the high level and middle level components, another for the adaptive mode transition control, one for the UAV model and the last one for a Monitor module used to save the signals involved in the simulation for posterior analysis.

Simulation Results

The hierarchical/intelligent control architecture was tested in simulation for the control of a Yamaha Rmax helicopter. The simulation includes only two modes (one for hover and the second for

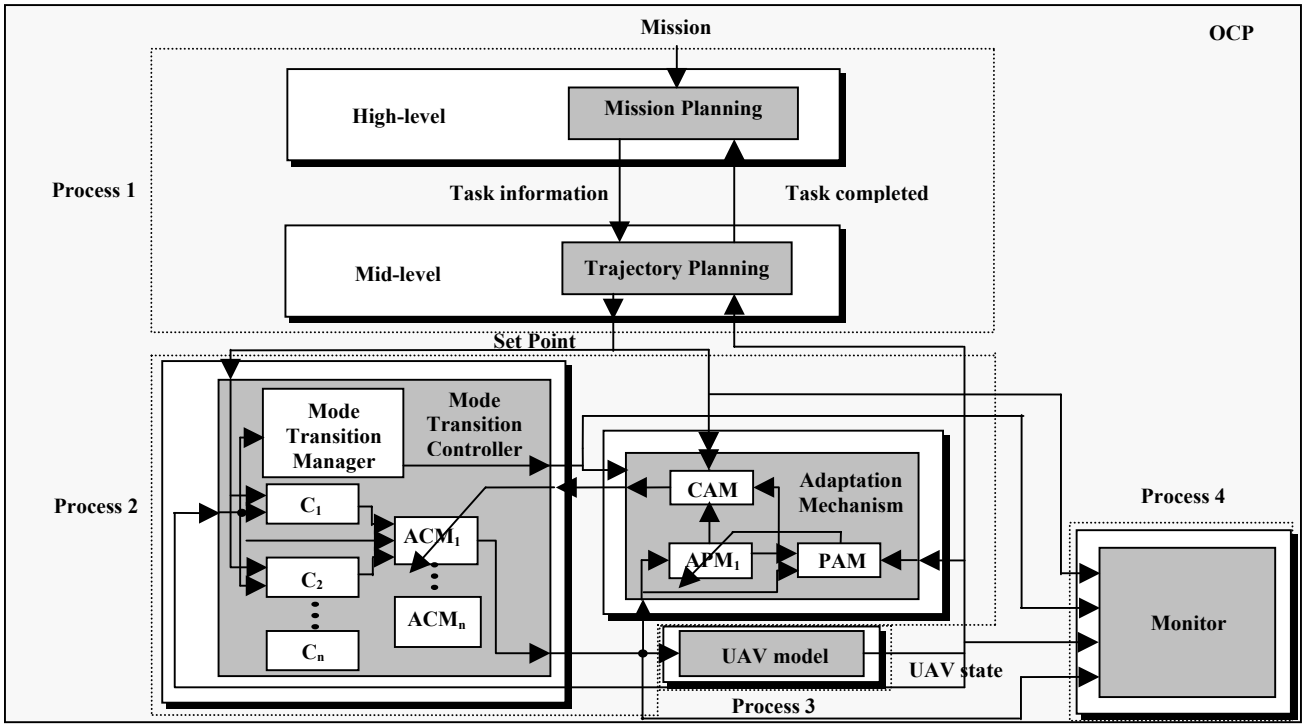

Figure 3. Adaptive Mode Transition Control On The OCP

forward flight at 50ft/sec) and one transition between these two modes. Simulation results are presented in Figure 4.

Values for the mean and maximum position and heading errors for simulations of one local controller, the mode transition control without adaptation, and the adaptive mode transition control are presented in Table 1. Notice that the mode transition control decreases the error with further decrease in the heading error for the adaptive mode transition control. The position errors in the last case increased slightly since the weights used in the optimization algorithm for the control adaptation mechanism were higher for the heading error, so the total weighted error decreased.

Table 1. Position and Heading Errors

Case	Mean Position Error (ft)	Max Position Error (ft)	Mean Heading Error (deg)	Max Heading Error (deg)
Local	0.89749	2.88952	4.04552	8.94898
MTC	0.77787	2.82648	1.09550	6.11610
AMTC	0.78754	2.82514	0.86399	5.35143

Conclusions

A hierarchical/intelligent control architecture for an unmanned aerial vehicle is proposed. The architecture is based on an adaptive mode transition control scheme, which entails new components in the middle level and high level to establish a mission and generate the set points for the corresponding trajectory. Simulation results are presented for the application of this architecture to the control of a rotary wing UAV. The results illustrate the effectiveness of the scheme. In the near future this architecture will be tested using a software-in-the-loop simulation of the vehicle, to be followed by a hardware-in-the-loop simulation, and finally a flight test.

Acknowledgements

This work is supported by the DARPA Software Enabled Control Program under Contracts #33615-98-C-1341 and#33615-99-C-1500, managed by the Air Force Research Laboratory (AFRL). We gratefully acknowledge DARPA's Software Enabled Control Program, AFRL, and Boeing Phantom Works for their continued support.

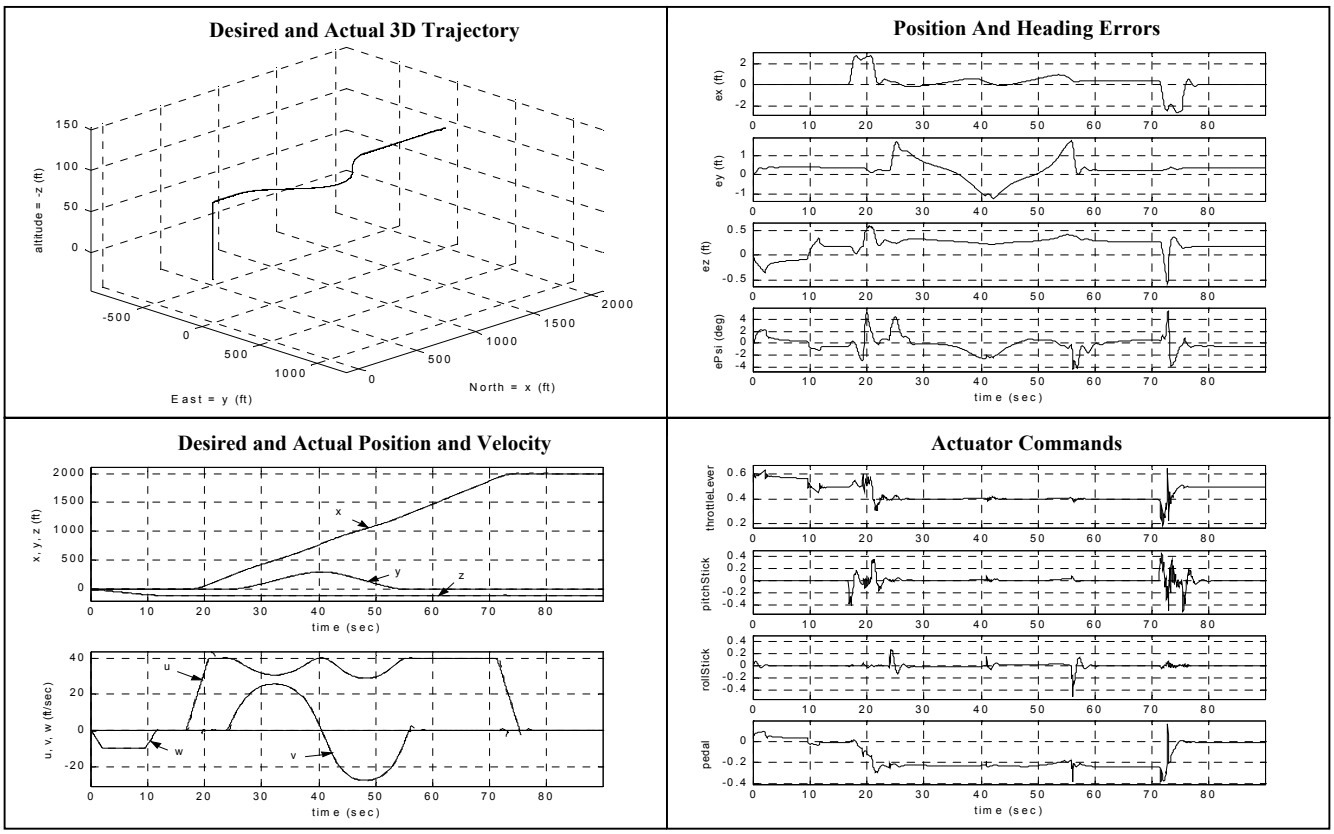

Figure 4. Simulation Results

References

[1] Heck, Bonnie, Linda Wills, and George Vachtsevanos, 2001, *Software Enabled Control: Background and Motivation*, Vol.5, Proceedings of American Control Conference, pp.3433-3438.

[2] Rufus, Freeman, George Vachtsevanos, and Bonnie Heck, 2002, *Real-time Adaptation of Mode Transition Controllers*, Journal of Guidance, Control, and Dynamics, Vol.25, No.1, pp.167-75.

[3] Rufus, Freeman, George Vachtsevanos, and Bonnie Heck, 2000, *Adaptive Mode Transition Control of Nonlinear Systems Using Fuzzy Neural Networks*, 8th IEEE Mediterranean Conference on Control and Automation, Patras, Greece.

[4] Rufus, Freeman, Bonnie Heck, and George Vachtsevanos, 2000, *Software-enabled Adaptive Mode Transition Control for Autonomous Unmanned Vehicles*, Proceedings 19th Digital Avionics Systems Conference, Vol. 1, pp. 1.E.1-1 - 1.E.1-8.

[5] Rufus, Freeman, 2000, *Intelligent approaches to mode transition control*, 2001, Ph. D. Thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA.

[6] Wills, Linda, Suresh Kannan, Sam Sander, Murat Guler, Bonnie Heck, J.V.R. Prasad, Daniel Schrage, and George Vachtsevanos, 2001, *An Open Platform for Reconfigurable Control*, Vol. 21, No. 3, IEEE Control Systems Magazine, pp. 49-64.

[7] Wills, Linda, Suresh Kannan, Bonnie Heck, George Vachtsevanos, C. Restrepo, Sam Sander, Daniel Schrage, and J.V.R. Prasad, 2000, *An Open Software Infrastructure for Reconfigurable Control Systems*, Vol.4, Proceedings of the 2000 American Control Conference, pp.2799-2803.

[8] Wills, Linda, S. Sander, S. Kannan, A. Kahn, J.V.R. Prasad, and D. Schrage, 2000, *An Open Control Platform for Reconfigurable, Distributed, Hierarchical Control Systems*, Vol. 1, Proceedings

19th Digital Avionics Systems Conferences, Philadelphia, PA, pp. 4D2/1 -4D2/8.

[9] Guler, Murat, Scott Clements, N. Kejriwal, Linda Wills, Bonnie Heck, and George Vachtsevanos, 2002, *Rapid Prototyping of Transition Management Code for Reconfigurable Control Systems*, Proceedings of the 13th IEEE International Workshop on Rapid Systems Prototyping (RSP), Darmstadt, Germany, pp. 76-83.

[10] Guler, Murat, Scott Clements, Linda Wills, Bonnie Heck, and George Vachtsevanos, 2003, *Generic Transition Management for Reconfigurable Hybrid Control Systems*, to appear IEEE Control Systems Magazine, 28 pages.