JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, December 2004

A Multi-Layer Autonomous Intelligent Control Architecture for Unmanned Aerial Vehicles

Jovan D. Boskovic,* Ravi Prasanth,† and Raman K. Mehra‡
Scientific Systems Company, Inc., Woburn, Massachusetts 01801

In the past several years there has been a lot of interest in the design of efficient autonomous intelligent controllers for Unmanned Aerial Vehicles (UAV). This is a highly complex and challenging problem since future UAVs will be expected to complete autonomously a wide variety of complex missions, and achieve performance comparable to that of manned vehicles. In this paper a four-layer autonomous intelligent control architecture for UAVs is described, and related issues are discussed. The architecture consists of the following layers: (i) Redundancy Management Layer that consists of the on-line Failure Detection and Identification (FDI) and robust feedback Adaptive Reconfigurable Controller (ARC); (ii) Autonomous Trajectory Generation (ATG) layer whose role is to fit feasible trajectories through the desired way-points in real time; (iii) Autonomous Path Planning (APP) layer that generates way-points on-line in response to a dynamically changing environment; and (iv) Autonomous Decision Making (ADM) layer whose role is to assess the available control authority after failures, and make mission-related decisions in near-real time. The main distinguishing feature of the architecture is that its layers are connected through the Achievable Dynamic Performance (ADP) calculation module which results in a system in which all the decisions are made based on the current available resources. Recent extensions of this architecture are also discussed and described in detail. At the end, a discussion is included on the Verification and Validation (V&V) of intelligent and adaptive control systems, and some recent results are presented.

Nomenclature

b_i	=	ith column of matrix B
p	=	Unknown parameter vector
r	=	Reference input vector
t	=	Variable denoting time
u	=	Control input vector
u_i	=	ith control input (ith entry of u)
$(u_i)_{\min}$	=	Lower bounds on the ith control input
$(u_i)_{\max}$	=	Upper bounds on the ith control input
$\bar{u}_i$	=	Upper bound on the magnitude of rate of change of ith control input
v	=	Auxiliary control input vector
w	=	Exogenous input vector
x	=	State vector of plant
x_m	=	State vector of reference model

This paper is part of the December Special Section on Intelligent Systems. Received 16 August 2004; revision received 12 November 2004; accepted for publication 19 November 2004.

*Intelligent & Autonomous Control Systems Group Leader, 500 W. Cummings Park, Suite 3000, jovan@ssci.com. AIAA Senior Member,
†Principal Research Scientist, 500 W. Cummings Park, Suite 3000, prasanth@ssci.com. AIAA Member.
‡President & CEO, 500 W. Cummings Park, Suite 3000, rkm@ssci.com. AIAA Member.

x_1 = Kinematic state vector
x_2 = Dynamic state vector
$\dot{x}$ = Derivative of state vector with respect to time
z = Persistent disturbance input
β = Relative scalar weight on auxiliary control input
η = Achievable dynamic performance variable
λ = Discrete-valued system mode (such as flight control mode)
μ = Costate vector
ρ_i = Risk potential for threat
Δt = Sampling time used for time discretization
A, B = State space matrices of linearized plant model
A_m, B_m = State space matrices of reference model
$I_{m\times m}$ = Identity matrix of size $m \times m$
J = Risk functional
M = Total number of threats
S_u = Set of admissible control input vectors
S_η = Achievable dynamic performance set

I. Introduction

Unmanned Air Vehicles (UAV) are becoming an integral part of future military forces. It is envisioned that future UAVs will perform autonomously complex tasks such as Intelligence, Surveillance and Reconnaissance (ISR), Close Air Support (CAS), Suppression of Enemy Air Defenses (SEAD), aerial refueling, and precision strike missions. The UAV formations will be required to achieve autonomous fault-tolerant and collision-free operation and conflict resolution when carrying out such missions. It is envisioned that ”swarms” of autonomous UAVs with an effective coordination strategy will lead to superior performance and efficient utilization of resources, and will achieve effective force superiority in a future battlefield. To achieve these objectives with minimum human intervention and in the presence of large external disturbances, different threats, flight-critical failures, and battle damage, there is a growing interest in developing highly efficient Autonomous Intelligent Flight Control Systems (AI-FCS) for both single and multiple UAVs, which is recognized to be a major enabling technology for their autonomous operation in dynamically changing environments.

There are numerous challenges facing the designer of such autonomous control systems. One of these is the situational awareness of the control system. This is closely related to the type of sensors that the vehicle is equipped with (radar, LIDAR, optical sensors, GPS). These sensors should provide the vehicle’s control system with the information on its location in space with respect to the known landmarks, such as those arising from the DTED maps, threats, obstacles, and other manned and unmanned aerial vehicles sharing the same theater of operation. Another important issue is autonomous control reconfiguration in the presence of subsystem and component failures, battle damage, and other upsets. Hence one of the objectives is to design a reliable and efficient on-line Failure Detection and Identification (FDI) and Adaptive Reconfigurable Control (ARC) system that can effectively compensate for severe failures and battle damage and assure vehicle’s survivability. The control system will also need to autonomously design and redesign trajectories that the UAV should follow. Hence another challenge is to design an efficient trajectory management system. One of the issues that arises in this context is to provide meaningful models of the vehicle’s post-failure/damage dynamics to the trajectory management system. The latter also needs to estimate the maximum performance that can be achieved post-failure/damage, and redesign the way-points and trajectories accordingly. In addition, the control system needs to make mission-related decisions autonomously. Another issue is that of communications with other manned and unmanned vehicles and the command center. Additional challenges include integration of the overall control system, its testing, and Verification and Validation (V&V).

In this paper some of these challenges will be discussed, and a hierarchical autonomous intelligent flight control system for UAVs, that is currently under development, will be described. Particular emphasis will be put on the concept of Achievable Dynamic Performance (ADP), i.e. the maximum performance that can be achieved under failures and disturbances, as the related measure is the main link between different levels of hierarchy. V&V of intelligent control systems will also be discussed in detail, and some recent results will be presented.

The main contribution of the paper is an architecture for autonomous intelligent control of UAVs in which the layers are connected through the Achievable Dynamic Performance (ADP) calculation module. This results in a system in which all the decisions are made based on an optimum use of the current available resources.

II. Issues in Autonomous Intelligent Control Design

In this section the desired functionality of an Autonomous Intelligent Flight Control System (AI-FCS) is described, and important issues that arise in its design and implementation are discussed.

A. Autonomous Intelligent Control

One of the first questions that arises in the context of intelligent control design is the definition of Autonomous Intelligent Control. One possible definition is given here: Autonomous intelligent control is execution of a given control strategy without human intervention and in an optimal manner, and capability to adapt autonomously and in a fast and efficient manner to a new set of circumstances by on-line sensing, information processing and control recon figuration. The difference between intelligent and adaptive control can be thought of as being based on the amount of uncertainty that can be handled by each approach. While adaptive control can compensate for small to moderate uncertainty, intelligent control can achieve the objectives in the presence of very large uncertainties arising in dynamically changing environments.

B. Autonomous Flight Control System for UAVs

A generalized schematic of an autonomous control system for UAVs that shows its desired functionality is given in Fig. 1. At the lowest level is the inner-loop controller, also referred to as the redundancy management system. The desired role of the inner-loop controller is to assure rapid stabilization of the overall system in the presence of failures, battle damage and state, control input and vehicle constraints, and improve accuracy of vehicle models through on-line learning. The inner-loop controller should therefore consist of several interconnected on-line subsystems including the Failure Detection and Identification (FDI) subsystem, Adaptive Reconfigurable Control (ARC) subsystem, control allocation subsystem and on-line learning and identification subsystem. This controller interprets and executes the commands generated by the outer-loop controller. The desired role of the latter, also referred to as the trajectory management system, is to react to unanticipated events by recon figuring path and trajectory to avoid pop-up threats or pursue targets of opportunity. Based on situational awareness and other sensory information, the decision-making layer is desired to make in near-real-time mission-related decisions including: (i) decisions whether (or not) to pursue a target of opportunity, (ii) decisions whether to continue, retask, or abort the mission, (iii) decisions based on available control authority after subsystem or component failure or damage. Related issues arising from the above architecture and its desired functionality are discussed below.

C. Autonomous Decision Making (ADM)

One of the major challenges in the design of autonomous control systems for UAVs is to devise suitable decision-making mechanisms and corresponding algorithms. The autonomous control system needs to have timely information about its internal state and external world, estimate accurately its available resources, and make rapidly corresponding decisions. For instance, in the case of subsystem or component failures, the available control authority in the post-failure vehicle may be lower than that in the nominal case. In such a case, the control system needs to compare the available resources with those needed to execute the original mission plan, and make a decision to continue, retask, or abort the mission.

D. Autonomous Path Planning (APP) and Autonomous Trajectory Generation (ATG)

Another important design challenge is to arrive at efficient algorithms for on-line generation and execution of a motion plan that enables the vehicle to move to a desired location and perform a given task, even while avoiding obstacles and radar detection. Given different way-points along a desired path, the objective of the Autonomous Trajectory Generation (ATG) system is to fit a feasible trajectory through the way-points, given the vehicle, terrain and control input constraints. Many of the trajectories can be calculated off line and stored. However, in the presence of pop-up threats, subsystem or component failures, and/or structural damage, the trajectory may need to be recon figured on-line to reflect the new environment, or the new achievable dynamics, or both. The control system will need to generate and execute the mission plan in near real-time and in an environment with a complex topology and with dynamically changing and uncertain components. Different motion planning techniques have been extensively applied in robotics,[34] and some of those have been applied in aerospace as well. Other techniques that have been used in aerospace include those based on the so-called probabilistic maps[35] and Voronoi diagrams.[26]

There are several available approaches to trajectory generation, including that based on the two-point boundary value problem,[47] the approach based on differential flatness and Linear Matrix Inequalities,[27,43] dynamic-programming-based approach,[28] and that based on Rapidly-exploring Random Trees (RRT).[35] The techniques differ in the required computing power, capability of near real-time operation, fast reconfiguration capability in the presence of changes in the environment, robustness to initial guesses, choice of optimization criteria, etc. There is a need to compare different techniques and arrive at a solution that is best suited for autonomous near real-time path planning and trajectory generation for UAVs. In addition, another important issue is that of providing a meaningful model to the ATG system post-failure. Severe failures or battle damage may substantially change the vehicle dynamics, and corresponding dynamic models may not be available. Hence some type of learning mechanism is needed to identify the model of the vehicle dynamics post-failure and use it for autonomous trajectory generation.

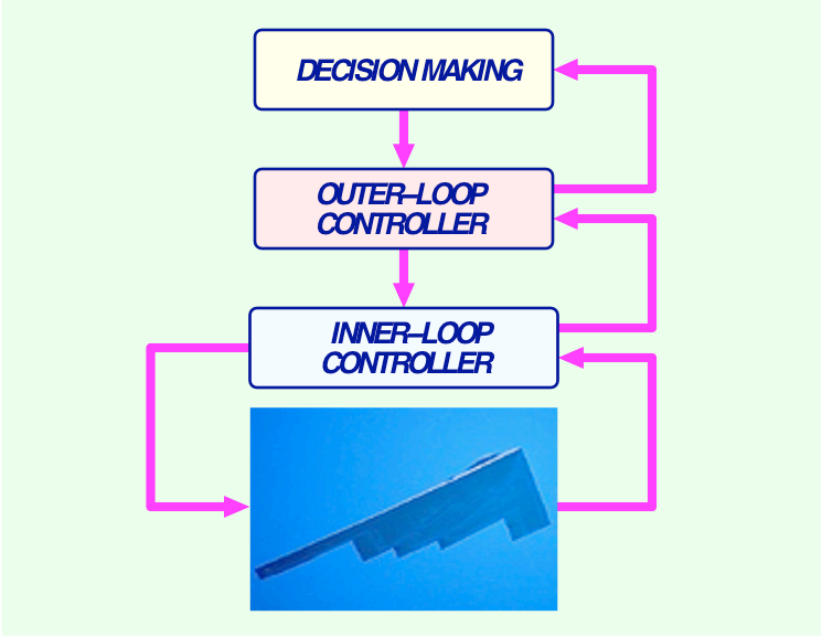

Fig. 1 Structure of the Autonomous Control System for UCAVs (courtesy of Scientific Systems Company, Inc.).

E. Fault-Tolerant and Reconfigurable Control

In the past several years there has been substantial progress in the area of fault-tolerant and reconfigurable control designs for both manned and unmanned aircraft.[1,3,8,16,19,22,23] The results so far have demonstrated the potential of the recon figuration techniques to maintain automatically the desired aircraft performance despite severe control effector failures and structural or battle damage. Several of those approaches have been extensively tested through piloted simulations,8 and even flight tested.[5,19] However, most of the proposed techniques are complex and it is not clear at all as to how to integrate them with the guidance and path-planning loops to achieve truly autonomous operation under different upsets, failures and unanticipated events. In the context of fault-tolerant and reconfigurable control, important issues include: (i) How to model different cases of critical failures and structural damage, but avoid a large number of models? (ii) How to integrate different on-line Failure Detection and Identification (FDI) and Adaptive Reconfigurable Control (ARC) algorithms to cover different types of upsets including sensor faults, control effector failures, and structural and battle damage? and (iii) How to change the Control Allocation Algorithms (CAA), arising in the context of overactuated aircraft, in the presence of failures, battle damage and other upsets?

F. Verification & Validation (V&V)

Increased autonomy of the future UAVs will require more extensive and thorough procedures for functional and software Verification and Validation (V&V) of their control system behavior in response to anticipated and unanticipated events or threats, subsystem and component failures, and battle damage. The on-line learning and adaptation associated with failure or battle damage accommodation, trajectory and path replanning, and on-line state

estimation and model identification in autonomous intelligent control systems makes the related V&V problem highly complex and often intractable. The issues arising in V&V of intelligent and adaptive flight control systems are discussed in detail in section VII.

G. Autonomous Intelligent Control of Multiple UAVs

In the context of multi-vehicle combat missions, integral components of the autonomous formation control scheme should include: (i) An autonomous path planning subsystem whose role is to generate feasible paths for the entire formation; (ii) An autonomous trajectory generation subsystem that generates desired temporal trajectories that can be followed by all vehicles in the formation; and (iii) An autonomous formation hold subsystem that initializes and maintains the formation throughout the mission. Given the complex nature of the corresponding tasks, the design of each of the above components can be truly formidable. The path planner should generate a path that avoids pop-up threats, and takes advantage of targets of opportunity whenever they arise. The trajectory generator should optimize the performance of the entire fleet while making sure that the commanded trajectories are within the performance limitations of all the vehicles. The formation hold autopilot should maintain the formation in a robust and collision-free manner, even in the face of large maneuvers of the entire formation. Most importantly, to allow flexible and efficient coordinated control of multiple vehicles, all these three components need to be designed and implemented in a decentralized fashion.

H. Quality of Information (QoI)

Each UAV's knowledge about itself and its environment can be grouped into private information that is obtained from on-board sensors (e.g. a proximity sensor), and communicated information that is obtained through a wireless network (e.g. time-division-multiple-access (TDMA) network such as Link-16). At each level of the hierarchical control structure, the information is processed to arrive at the appropriate decisions, including striking targets. A key issue is the quality of information (QoI) used in these decision making processes. Intuitively, a piece of information is said to be of high quality if it is closer to the truth. This notion of QoI has been formalized using Kullback-Leibler information distance and applied to autonomous decision-making.[42] An intelligent control system determines the QoI of the received information and, if the QoI is above a threshold, uses the data to carry out its mission. This provides a higher level of reliable autonomous operation against communication faults (jamming, packet drops, etc.), and malicious tampering of data.

III. Autonomous Intelligent Controller for UAVs

In this section one possible approach to the autonomous intelligent control design for UAVs is described. A hierarchical control architecture is presented next, followed by a description of the basic approach to FDI-ARC design and V&V.

Several existing architectures for autonomous intelligent control for UAVs are discussed first, and the main distinguishing features of the architecture proposed in this paper are discussed.

Hierarchical Control Structure. While the generalized architecture from Figure 1 has been widely accepted in the field, the existing hierarchical controllers differ in the ways in which each of the blocks is implemented. For instance, the Decision Making Block is often replaced with a mission-level layer where pre-calculated paths and trajectories are stored and commanded to the vehicle to follow. This block can also carry out path or trajectory replanning in response to the changes in the environment. The outer-loop controller can be implemented as two separate blocks [Autonomous Path Planning (APP) and Autonomous Trajectory Generation (ATG)], or as a single Autonomous Motion Planning block.[28] Additional blocks may be needed to implement a specific path planning or trajectory generation algorithm.[2]

A. Hierarchical Autonomous Intelligent Controller

To address some of the issues discussed in the previous section, a hierarchical architecture for the Autonomous Intelligent Flight Control System (AI-FCS), shown in Figure 2, is proposed, and several of its elements are defined and studied. Different layers and blocks are described below.

Achievable Dynamic Performance (ADP). The main distinguishing feature of the proposed Hierarchical Autonomous Intelligent Controller is the Achievable Dynamics Performance (ADP) block. ADP is de fined as the maximum performance that the vehicle can achieve under different faults, failures, and external disturbances in a dynamically varying environment. For instance, in the case of wing damage in a fixed-wing UAV, the damage may not only render one or more control effectors ineffective, but will also result in aerodynamic perturbations due to asymmetry that will severely limit the tasks that the vehicle can achieve post-damage. In the proposed architecture,

an ADP measure is calculated on-line at the inner-loop control level, and passed on to the higher hierarchical levels that make appropriate changes to reflect the new lowered capabilities of the vehicle. The ADP concept is described in detail in the following section.

Level 4: Decision-Making Layer. This layer has the information about the overall mission objectives and constraints. This information, in conjunction with the sensory and ADP information and situational awareness, is used to make appropriate decisions as trade-offs between the mission success and vehicle survivability. This layer is responsible for collision avoidance, conflict resolution, mission retasking, and goal reassessment.

Level 3: Path Planning Layer. The role of this layer is to generate the motion plan for the overall mission, and compute spatial and other constraints needed for the design of the desired trajectories. Many of the routes and constraints can be computed off-line to cover different situations, including the nominal case and a set of anticipated events, and stored in memory. The constraints are computed in the form of safe set boundaries around the way-points. An example of way-points and constraints in the nominal case (i.e. the case without disturbances, failures, upsets, and unanticipated events) is shown in Figure 3(a). This layer also constantly monitors current ADP measure, vehicle dynamics and external events, and, if necessary, recon figures/changes the precomputed path. This is illustrated in Figure 3(b) in the case of a pop-up threat. As seen from the figure, the way-points and the safe set boundaries are recomputed on-line to avoid the threat.

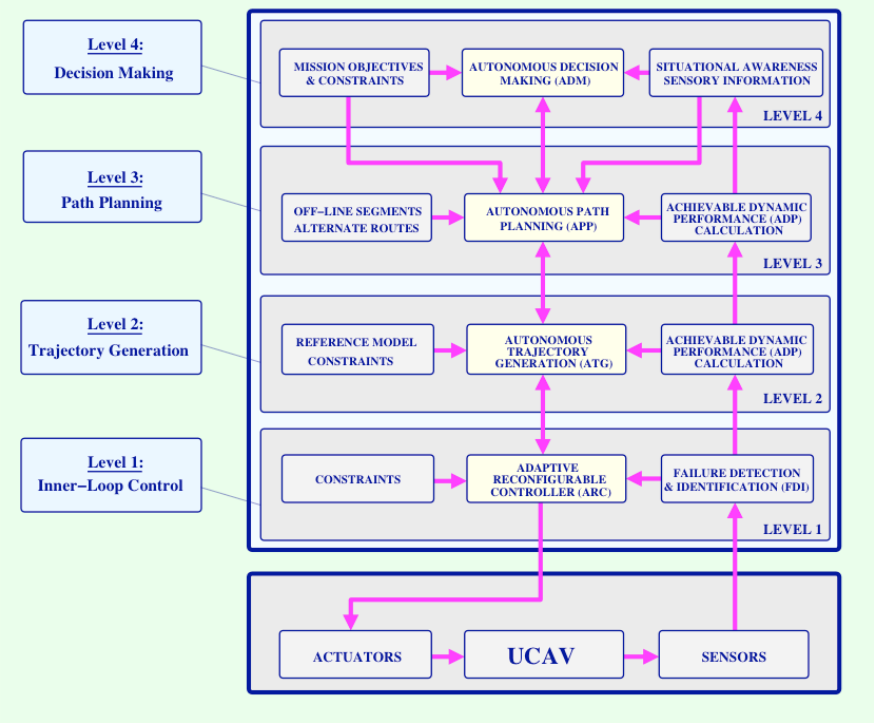

Fig. 2 Structure of the Hierarchical Controller (courtesy of Scientific Systems Company, Inc.).

In this context a simplified path planning algorithm applicable to a two-dimensional case has been recently developed. The design procedure is illustrated in Figure 4. Beginning at the top left corner of Figure 4, a coordinate transformation is applied so that the threat region becomes a unit circle centered at the origin (z-domain). Then, a conformal transformation is applied to interchange the threat region's interior and exterior (ω-domain). The problem is to plot a course between the current spatial state and the desired final spatial state that lies inside the unit circle. Since the conformal transformation takes ∞ to 0, trajectories that come close to zero in the ω-domain will be mapped back into unbounded trajectories. Therefore, a trajectory that is close to the circle is chosen (see Figure 4). Another possibility is to put way-points within the unit circle, map them back and generate trajectory between the way-points. If there are multiple threats as shown in the bottom left corner of Figure 4, then an ellipsoid of minimum volume (area) that contains all the threats is computed first to reduce the situation to the one just discussed.

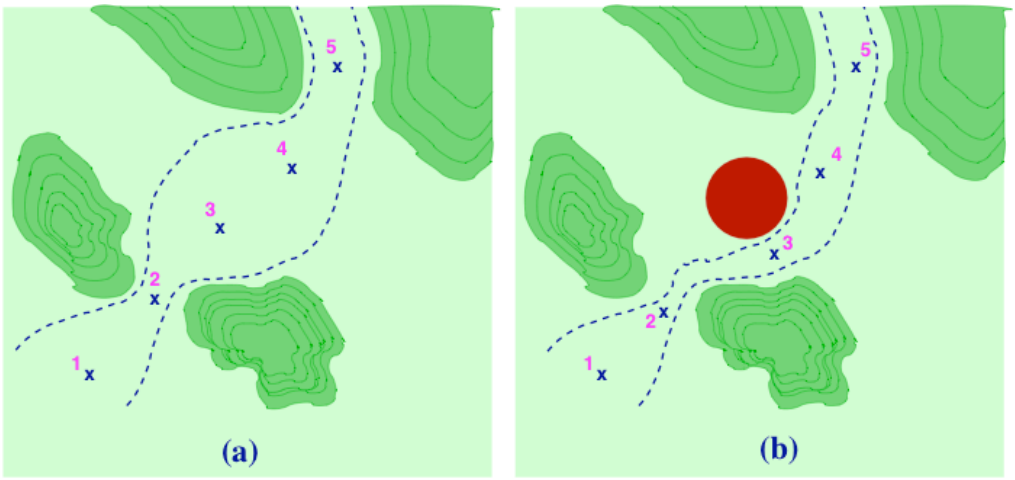

Fig. 3 Autonomous path planning: (a) Nominal case; (b) On-line path reconfiguration due to a pop-up threat (courtesy of Scientific Systems Company, Inc.).

It may be more reasonable in situations such as the one shown in Figure 5 to view threats as regions of varying risk rather than as regions to be avoided completely. To formulate trajectory generation problem in this case, let us attach to each trajectory $x(t),\ t \in [0,T]$ a risk value:

$$J(x) = \sum_{i=1}^{M} \int_0^T \rho_i(x(t))dt$$

where ρ_i is a differentiable positive function describing the risk potential associated with the ith threat, and M is the total number of threats known to the decision-maker. Typically, the functions ρ_i roll-off away from the threat region. The trajectory generation problem is to find a state trajectory from the current state to the final state that minimizes the risk functional J subject to system dynamics and constraints. This is a problem in variational calculus and is often very difficult to solve.[20] The computational difficulties associated with a direct solution of the risk minimization problem can be alleviated by decomposing the problem into a Level-3 path planning problem followed by a Level-2 trajectory generation as described next. Note that the equations of motion (EOMs) of flight vehicles have the following general form*:

$$\dot{x}_1 = f_1(x_2) \tag{1}$$

$$\dot{x}_2 = f_2(x_1, x_2, u) \tag{2}$$

where x_1 is a vector of kinematic state variables, x_2 is a vector of dynamic state variables and u is a vector of control inputs. Examples of kinematic variables are inertial position and Euler angles, and examples of dynamic variables are airspeed, side-slip angle, rotational rates, and flight path angle. It is reasonable to assume that the risk potential ρ_i is a function only of the kinematic states so that:

$$J(x) = J(x_1)$$

i.e, the risk value associated with a trajectory depends only on the path. Now, if J is a constant for all paths, then a path given by the solution of

*This form is valid under some assumptions. For example, dependence of air density on altitude, which is a state variable is neglected.

$$\dot{x}_1 = -x_1 + x_1^f \quad x_1(0) = x_1^0$$

where x_1^0 and x_1^f are the initial and final kinematic states respectively, will surface for all practical purposes. Moreover, since kinematics is invertible, one has that

$$x_2^{command} = f_1^{-1}(\dot{x}_1) = f_1^{-1} = \left(-x_1 + x_1^f\right)$$

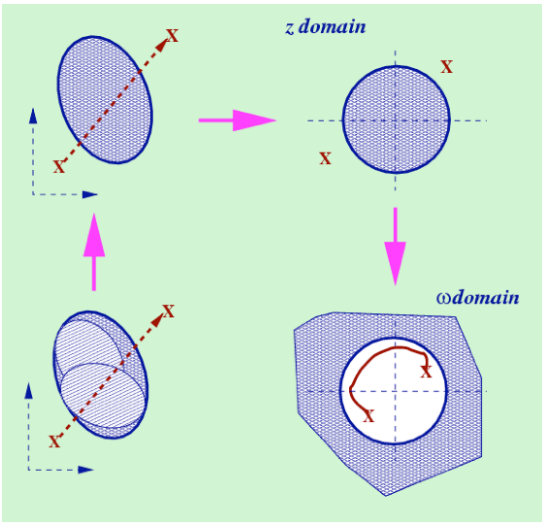

Fig. 4 Trajectory generation under pop-up threats -2D case (courtesy of Scientific Systems Company, Inc.).

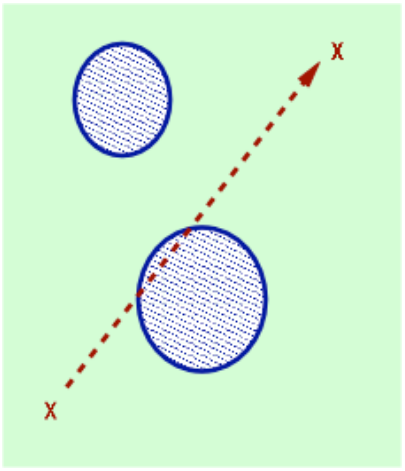

Fig. 5 Trajectory generation under pop-up threats -case for risk minimization.

as the command for the lower-level trajectory generator. The equation for $\dot{x}_1$ can be generalized to include a reference model and, in fact, such a generalization is needed to accommodate the nonlinear nature of *J*. Furthermore, it is desirable to have forcing function $g\left(x_1^f - x_1\right)$ to be somewhat uniform in magnitude throughout the domain and not increasing as is the case with a linear function above. With these observations, the following variational path planning problem is proposed:

$$\min_{v}\left(\beta\int_0^T v(t)'v(t)dt+\sum_{i=1}^{M}\int_0^T \rho_i\left(x_1(t)\right)dt\right) \tag{3}$$

$$\text{subject to } \dot{x}_1 = g\left(x_1^f - x_1\right) + v_1, \quad x_1(0) = x_1^0, x_1(T) = x_1^f \tag{4}$$

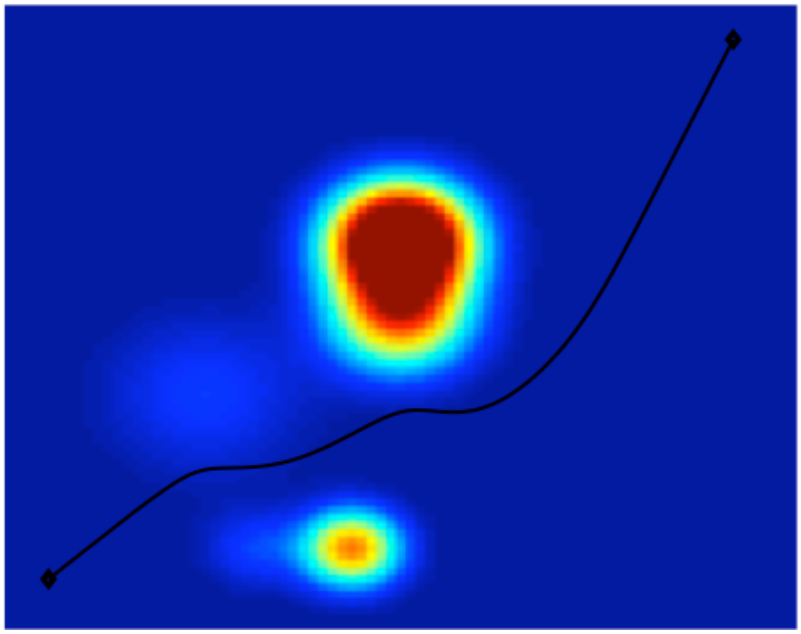

Fig. 6 Path planning using a combination of risk minimization and reference model tracking.

where v is an auxiliary control input, T is an unspecified final time and g is a differentiable positive function with properties mentioned above. A suitable candidate for g is

$$g(z) = \left(c_1 e^{-c_2 z'z}\right)z$$

where $c_1 > 0$ and $c_2 > 0$. Using standard variational principles,[20] one obtains

$$\dot{x}_1 = g\left(x_1^f - x_1\right) - \frac{\mu}{\beta} \tag{5}$$

$$\dot{\mu} = -\frac{\partial g\left(x_1^f - x_1\right)}{\partial x_1}\mu - \sum_{i=1}^{M}\frac{\partial \rho_i(x_1)}{\partial x_1} \tag{6}$$

as necessary conditions for stationary solution. The above equations should now be simulated to obtain the path from initial state x_1^0 to x_f^0. Figure 6 shows a path computed using this method.

It should be noted that the above described method does not always give a path along which the risk is close to its minimal value. This is because of the g-dependent terms in (5-6). On the other hand, without these terms, the variational solution as presented above may not provide a path that ends in the specified final state because the initial and final co-state conditions are not searched for. The advantage of this method, in addition to its computational speed, is that the existence of a path is guaranteed by choosing g properly. Homotopy or pheromone trail methods can then be applied to obtain better solutions.

Level 2: Trajectory Generation Layer. The role of this layer is to fit a feasible trajectory through the way-points even while satisfying the state, control input, and spatial constraints. Trajectory generation is commonly based on minimization of a given criterion (e.g. time between the way points, fuel consumption, or low exposure to known stationary threats), and can be generated either on-line or off-line. In the case of failures, upsets, or other anticipated or unanticipated events, the path planning layer automatically recon figures the desired path by modifying the way-points, while the trajectory generation layer fits a feasible trajectory that is achievable under the circumstances. To simplify the on-line trajectory re-design, the ATG system may be equipped with a Motion Primitives Library (MPL) that contains the so-called trim trajectories and maneuvers, de fined as transitions between the trim trajectories. The role of this layer is illustrated in Figure 7. In the nominal case, Figure 7(a), the trajectory generation layer fits a feasible trajectory by minimizing a given criterion (for instance fuel consumption), depicted by a solid line, between the way-points, In the case of a sudden pop-up threat (e.g. a previously undetected SAM site), the trajectory generation layer fits a new feasible trajectory. The new trajectory will depend on the time instant when the information about the new way-points and constraints was received by the trajectory generation layer. In the case from the figure it is assumed that this information was received at the second way-point. If, in addition to the pop-up threat, there is an upset condition, the vehicle cannot achieve the same performance (e.g. acceleration) as in the no-upset case, and the way-points and the desired trajectory need to be modified accordingly in order to avoid the unsafe zone with the available ADP. An initial discussion on on-line trajectory generation in the context of the architecture from Figure 2 is given in Ref. 43.

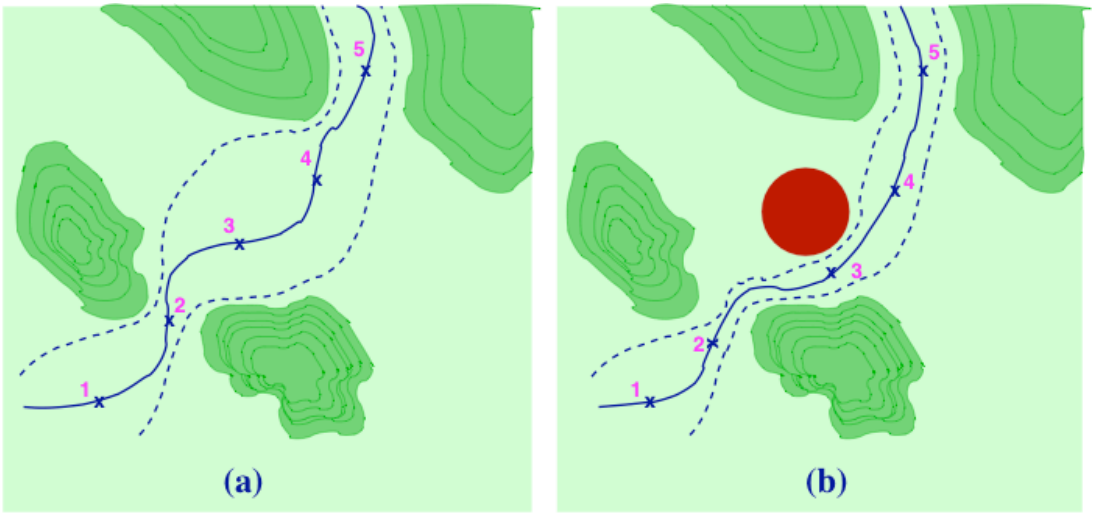

Fig. 7 Autonomous trajectory generation: (a) Nominal case; (b) On-line trajectory reconfiguration due to a pop-up threat (courtesy of Scientific Systems Company, Inc.).

Level 1: Redundancy Management Layer. The role of this layer is to ensure accurate following of the desired trajectory in the presence of different disturbances, failures and upsets. This layer is shown in Figure 8, and includes robust on-line FDI-ARC system that detects and identifies different failures, battle damage and disturbances, and recon figures the controller accordingly. The control allocation algorithm associated with the FDI-ARC system is designed to optimize redundancy management in the presence of control input constraints and varying available control authority. Previous R&D in the area of Failure Detection and Identification (FDI) and Adaptive Reconfigurable Control (ARC) has resulted in the development of several unique technologies, and relevant results are described in Refs. 8, 11, 12, 16. Many of the results on FDI-ARC design in the presence of flight-critical failures and battle damage are currently being extended and integrated within a hierarchical control architecture. The FDI-ARC techniques developed in this context are described next.

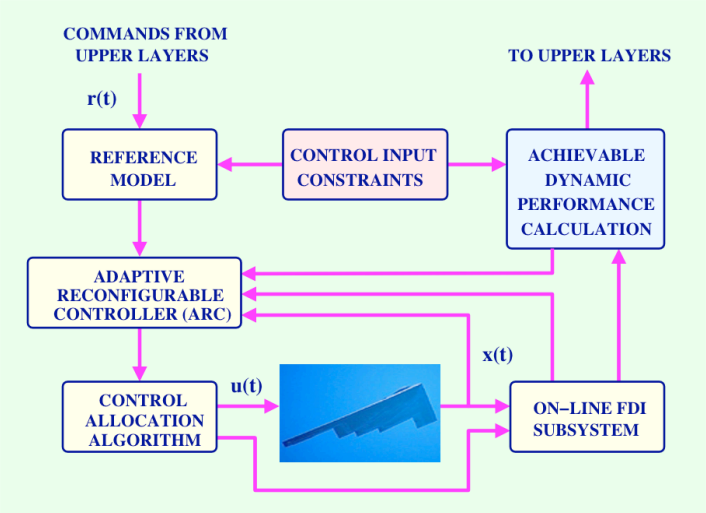

Fig. 8 Detailed Structure of the Level 1 Controller (courtesy of Scientific Systems Company, Inc.).

B. Failure Detection, Identification and Recon figuration (FDIR)

In this section two techniques for fast and accurate FDIR of flight-critical failures are described •one based on the concept of Multiple Models, Switching and Tuning (MMST), and the other one based on the Decentralized FDIR.

Multiple Models, Switching and Tuning (MMST). One promising approach to on-line FDI-ARC is based on the MMST concept developed by Narendra and coworkers in Ref. 36. This approach has been extended to the FDIR framework and an efficient FDI-ARC techniques have been developed.[12,18]

The MMST concept is illustrated in Figure 9. In the context of FDI-ARC, the basic idea is to represent different failure scenarios using corresponding on-line observers. The observers are used to find the one closest to the current operating regime and switch to the corresponding controller. Such an approach has been demonstrated as an efficient tool for FDI-ARC in the presence of different control effector failures in overactuated aircraft.[17,18] Its main feature is that is assures the stability of the overall system and guarantees that, in the presence of unknown failure, the FDI-ARC scheme will switch to the right controller thus assuring asymptotic convergence of the tracking error to zero.

A Decentralized FDIR Scheme. One of the main disadvantages of the above technique is that it is not well suited for FDIR in the presence of multiple simultaneous control effector failures. The main reason is that, for such failures, accurate models are needed that cover all possible combinations of failures, which results in a prohibitively large number of on-line observers. In addition, almost all approaches are highly centralized which results in a highly complex hybrid system. For instance, a centralized FDIR system based on the MMST technique and applied in the context of flight control is shown in Figure 10. In Refs. 11 and 14 a new Decentralized Failure Detection, Identification and Reconfiguration (FDIR) approach that is well suited for FDIR in the presence of multiple simultaneous control effector failures is discussed. The structure of the proposed Decentralized FDIR scheme is shown in Figure 11. It is seen that, in this case, the scheme is based on the local FDI observers. The Decentralized FDIR scheme has been recently evaluated through high-fidelity and piloted simulations at Boeing Phantom Works achieving excellent results for the F/A-18 aircraft in the presence of severe flight critical control effector failures.[8]

IV. Related Hierarchical Architectures

To integrate different hierarchical layers with situational awareness sensors within a comprehensive architecture, a new Multi-layer Architecture for Trajectory Replanning and Intelligent plan eXecution (MATRIX) system has been recently proposed.45 The main role of the MATRIX system is to integrate threat detection algorithms based on Interacting Multiple Models (IMM) with online path planning and trajectory generation within an effective multi-layer architecture for pop-up threat avoidance under subsystem and component faults and failures. An extension of the MATRIX architecture has been recently proposed. The architecture is referred to as the Integrated Motion Planning, Awareness and Control Technology (IMPACT) system, and will integrate vision-based pop-up threat detection with on-line motion planning for aggressive maneuvering to achieve mission objectives for UAVs under different threats and dynamic changes in the environment. The IMPACT architecture is shown in Figure 13 and expands the MATRIX architecture from Figure 12 by including a Motion Primitives Library (MPL) for aggressive

maneuvering and combining its elements with the Rapidly-exploring Random Trees (RRT) algorithm in the context of nonlinear vehicle dynamics. The proposed approach also extends related work on the ViSTA system (Visual System for Threat Awareness) under the DARPA Software-Enabled Control (SEC) Program[9] where the objective is to use stereo vision, Sarnoff's ACADIA board, advanced perceptual organization techniques and graph theory to design vision processing algorithms to detect the size and position of potential threats.

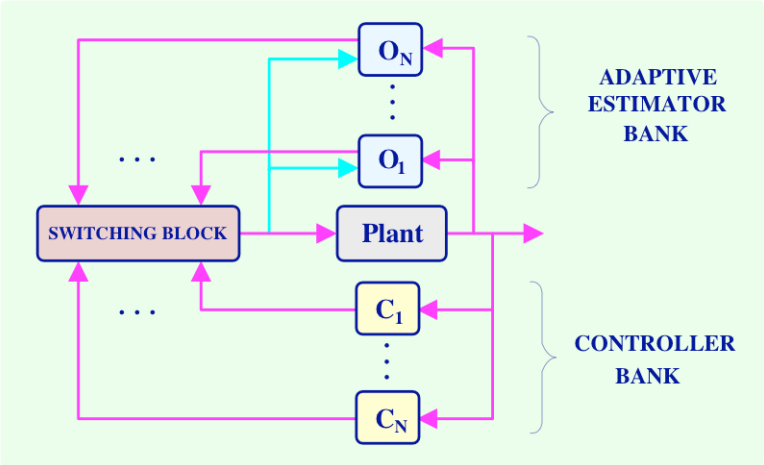

Fig. 9 Structure of the Multiple Model-Based Controller: Outputs of the parallel observers O_1 , O_2, ... O_N are used to find that closest in some sense to the current plant dynamics, and switch to the corresponding controller (courtesy of Scientific Systems Company, Inc.).

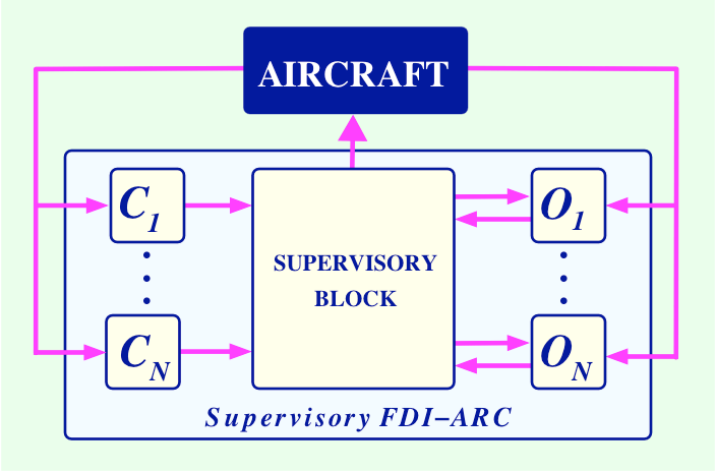

Fig. 10 The Decentralized MMST-based Scheme (courtesy of Scientific Systems Company, Inc.).

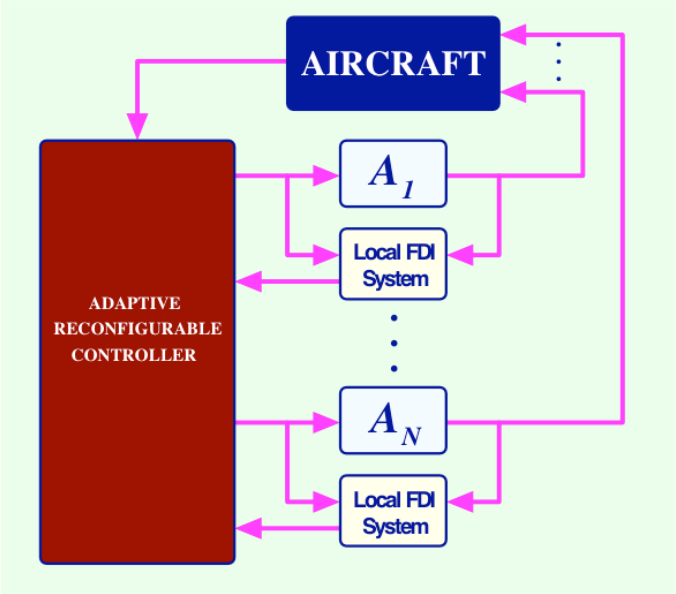

Fig. 11 Decentralized FDIR Scheme (courtesy of Scientific Systems Company, Inc.).

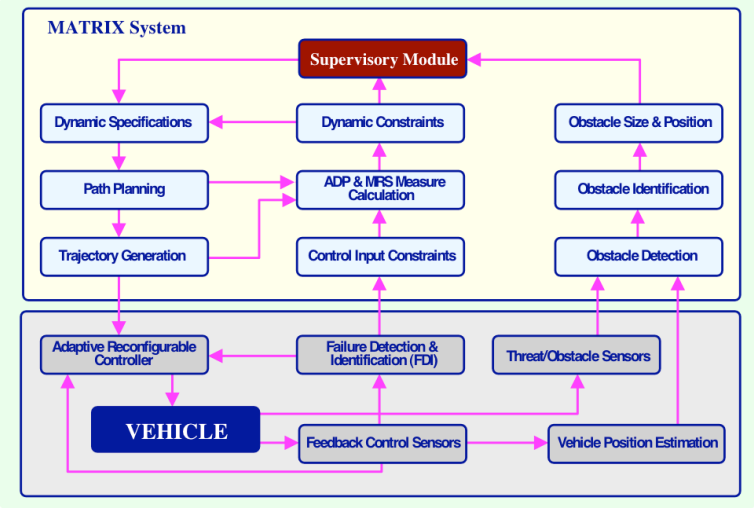

Fig. 12 Structure of the multi-layer architecture for trajectory replanning and intelligent plan execution (MATRIX) (ADP – achievable dynamic performance, MRS – maximum reachable sets) (courtesy of Scientific Systems Company, Inc.).

The major differences between the MATRIX/ViSTA and IMPACT architectures are as follows: (i) The MATRIX architecture is well suited for slow[9] to moderately aggressive maneuvers,[45] while the main focus under the IMPACT architecture is on aggressive maneuvers. (ii) The VISTA system is based on perceptual organization, graph theory and stereo vision. This approach is well suited for large vehicles that can provide a wide stereo baseline for improved range resolution. The IMPACT vision processing shares the same foundation, but is based on optical flow measurements, which are more appropriate for smaller vehicles. One of the major differences between the RRT motion planner being developed under the MATRIX project, and the one under the IMPACT architecture is that the former will be developed for a class of linear models of vehicle dynamics, while the latter will be based on nonlinear vehicle dynamics and the library of motion primitives.

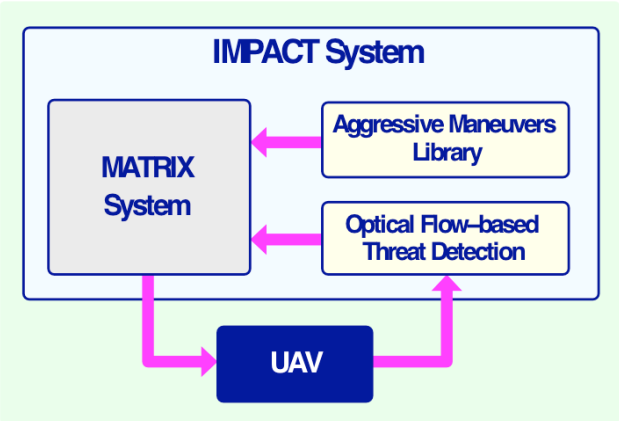

Fig. 13 Structure of the integrated motion planning, awareness and control technology (IMPACT) system (courtesy of Scientific Systems Company, Inc.).

V. Achievable Dynamic Performance (ADP)

In autonomous intelligent control of UAVs, the reconfigurable flight controller compensates for the effect disturbances, structural damage or subsystem failures that occur in the course of the mission. In such cases, a portion of the available Control Input Redundancy (CIR) is used to compensate for these effects through control recon figuration, which in turn results in lower available CIR. In critical cases, this may lead to situations in which the vehicle cannot achieve desired performance in all segments of the mission. For instance, maximum speed of the vehicle's response after control reconfiguration may be too low to avoid a missile, even though the "healthy" vehicle could readily achieve this.

An important concept in this context is that of the Achievable Dynamic Performance (ADP), i.e. the maximum performance, de fined in a suitable space, that the vehicle can achieve under different circumstances. A suitable ADP measure appears to be the ADP set that de fines all points that can be achieved with available control authority.

In order to calculate the ADP set immediately following failures, structural damage, or effects of disturbances, the ADP subsystem needs to receive on-line information from the Failure Detection and Identification (FDI) subsystem.

The ADP measure is critically important for determining whether or not the vehicle is capable of continuing and completing the mission. The corresponding decisions are made by the Autonomous Decision Making (ADM) subsystem described briefly in the previous section.

The main idea behind the approach to the ADP discussed here is to convert the dynamic tracking problem into a static one and solve it either on-line or off-line using suitable constrained optimization techniques. For instance, if the desired trajectory is known in advance and if there are no disturbances and/or failures or other upsets, the ADP set can be calculated off-line for different segments of a mission. Since compensation for different failures, upsets and disturbances results in lower ADP, the proposed online ADP subsystem calculates the maximum performance that can be achieved under dynamically changing circumstances.

The ADP problem will be considered next, and the manner in which it depends on the FDI information will be discussed.

The Tracking Problem. Let the plant dynamics be described by

$$\dot{x} = f(x) + Bu \tag{7}$$

where x and u are respectively n and m-vectors and f is a smooth vector function, and let $m > n$, i.e. there are more inputs than the variables to be controlled. It is assumed that the control effectors are subject to both position and rate limits, i.e. $u \in S_u = \left\{u : (u_i)_{\min} \le u \le (u_i)_{\max}, |\dot{u}_i| \le \bar{u}_i, i = 1,2,...,m\right\}$, Let the objective be to design a $u(t)$ such that the error $x(t)$ - $x_m(t)$ asymptotically converges to zero, where x_m is a state vector of a reference model specifying the desired performance of the plant:

$$\dot{x}_m = A_m x_m + B_m r \tag{8}$$

where A_m is an asymptotically stable matrix, and r is a smooth bounded function of time. Then, for $x(0) = x_m(0)$, the tracking problem under input constraints can be converted to a static problem for every t: *Minimize* $\varepsilon^T Q \varepsilon$, where $\varepsilon = Bu - \eta$, *subject to the constraints* $u \in S_u$, where

$$\eta(t) = -f\left(x(t)\right) + A_m x_m(t) + B_m r(t) \tag{9}$$

The ADP problem can be studied analytically in terms of a static equation of the form

$$Bu = \eta,$$

where $u \in S_u$ and η is defined above.

A. The ADP Problem in the Ideal Case

Find the set $S_\eta = \{\eta : \eta = Bu, u \in S_u\}$ that can be achieved with u $\in S_u$. In other words, the objective is to find the largest set S_η that can be achieved with available control authority. Once this set is found, the problem reduces to assuring that the actual $\eta(t)$ is within the set for all time.

The problem of finding the set S_η can be solved using Linear Matrix Inequalities (LMI) tool in Matlab, upon converting the constraint $u \in S_u$ into an LMI as follows.

The derivative of u can be approximated as

$$\dot{u}_i \cong \frac{u_i(k) - u_i(k-1)}{\Delta t}, \quad i = 1, 2, \ldots, m, \quad k = 1, 2, \ldots \tag{10}$$

where $t = k\Delta t$, and Δt denotes the sampling period.

Hence from $u \in S_u$ one has:

$$u_i(k-1) - \bar{u}_i \Delta t \le u_i(k) \le u_i(k-1) + \bar{u}_i \Delta t \tag{11}$$

This set of inequalities can be represented as an LMI of the form $C_1 u(k) \le d_1(k\text{-}1)$, where

$$C_1 = \begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix}, \quad d_1(k-1) = \begin{bmatrix} u_1(k-1) - \bar{u}_1 \Delta t \\ \vdots \\ u_m(k-1) - \bar{u}_m \Delta t \\ -u_1(k-1) - \bar{u}_1 \Delta t \\ \vdots \\ -u_m(k-1) - \bar{u}_m \Delta t \end{bmatrix} \tag{12}$$

Further, position limits can be represented as an LMI of the form $C_2 u \le d_2$, where

$$C_2 = \begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix}, \quad d_2 = \begin{bmatrix} (u_1)_{max} \\ \vdots \\ (u_m)_{max} \\ -(u_1)_{min} \\ \vdots \\ -(u_m)_{min} \end{bmatrix} \tag{13}$$

Hence one has

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} u(k) \le \begin{bmatrix} d_1(k-1) \\ d_2 \end{bmatrix}, k = 1,2... \tag{14}$$

In the following paragraphs, in order to illustrate the ADP concept in simple terms, it will be assumed that there are only position limits on the control effectors.

B. The ADP Problem in the Presence of Control Effector Failures

1) Hardover Failures. In the case of hardover failure of the *i*th control effector, the objective is to find the set S_η from the equation:

$$b_1u_1 + b_2u_2 + ... + b_{i-1}\,u_{i-1} + b_{i+1}u_{i+1} + ... + b_mu_m = \eta - b_i(u_i)_{max}, \tag{15}$$

where b_j is the *j*th column vector of B from Eq. (10), subject to the inequality constraints $(u_i)_{min} \le u_i \le (u_i)_{max}$.

2) Lock-in-Place Failures. The objective is the same in the case of lock-in-place of the *i*th effector, except that (ui)max is substituted by ui, i.e. the value at which the effector has frozen.

3) Loss-of-Effectiveness Failures. In the case of loss-of-effectiveness failure of the *i*th control effector, the objective is to find the set S_η from the equation:

$$b_1k_1u_1 + b_2k_2u_2 + \ldots + b_ik_iu_i + ... + b_m\,k_mu_m = \eta \tag{16}$$

subject to the inequality constraints $(u_i)_{min} \le u_i \le (u_i)_{max}$, where $k_j = 1, j = 1, 2, ..., m, j \ne i$, and $k_i = \bar{k}_i \in [0,1)$.

C. The ADP Problem in the Presence of Persistent External Disturbances

Let Eq. (7) be of the form:

$$\dot{x}_2 = f(x) + Bu + z \tag{17}$$

where z denotes a vector of bounded external disturbances such that $z(t) \le \bar{z}$ for all time. Then the ADP problem is the following: Find the set S_η from the equation

$$b_1u_1 + b_2u_2 + ... + b_mu_m = \eta - \bar{z} \tag{18}$$

where $\bar{z}$ is either a measured or an estimated value, subject to the inequality constraints $(u_i)_{min} \le (u_i) \le (u_i)_{max}$. Example: Let $m = 3$ and $n = 2$. In this case, from Eq. (10) it follows that $b_1u_1 + b_2u_2 + b_3u_3 = \eta$, where b_i and η are 2-vectors. Let $(u_i)_{max} = -(u_i)_{min} = 1$, $b_1 = [2\ 1]^T$, $b_2 = [1\ 1]^T$, and $b_3 = [0\ 1]^T$. The corresponding ADP set for the case is calculated using a standard convex hull algorithm in Matlab, and shown in the left part of Figure 14.

The case when the ADP set changes due to failures, disturbances and other upsets will be considered next. For instance, if the first control effector locks in place at the value $\bar{u}_i$, the equality constraint is now of the form $b_2u_2 + b_3u_3 = \eta - b_1\bar{u}_i$. The failures of the other effectors can be studied in the same way. For all values of $\bar{u}_i$ between [-1, 1], the ADP set changes as shown in Figures 14 and 15. It is seen that, in each case, the ADP sets are of the same size for all values of $\bar{u}_i$. This is due to the fact that there are only two "healthy" control effectors, while the location of the ADP set depends on the value at which the failed effector has frozen.

In the case of loss-of-effectiveness, the coefficients k_i multiplying each control input change from [0, 1]. The resulting changes of the ADP sets are shown in Figure 16. It is seen that the ADP set in the case of $k_i = 0$ coincides with that for the case of LIP failure and $\bar{u}_i = 0$.

As seen from the above example, the ADP set can change substantially due to a failure, which may limit the set of UAV commands that can be followed. The above problems emphasize the importance of fast and accurate FDI and disturbance estimation for effective on-line calculation of the ADP set.

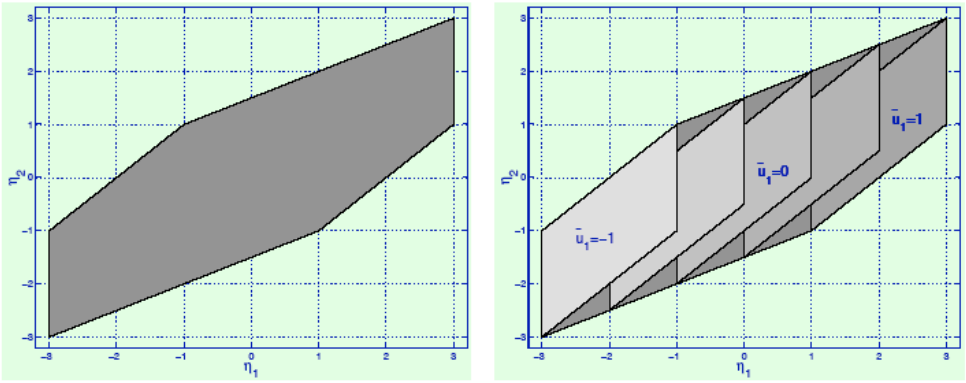

Fig. 14 Nominal ADP set (left) and changes in the ADP set due to a hard-over failure control effector u_1 (right).

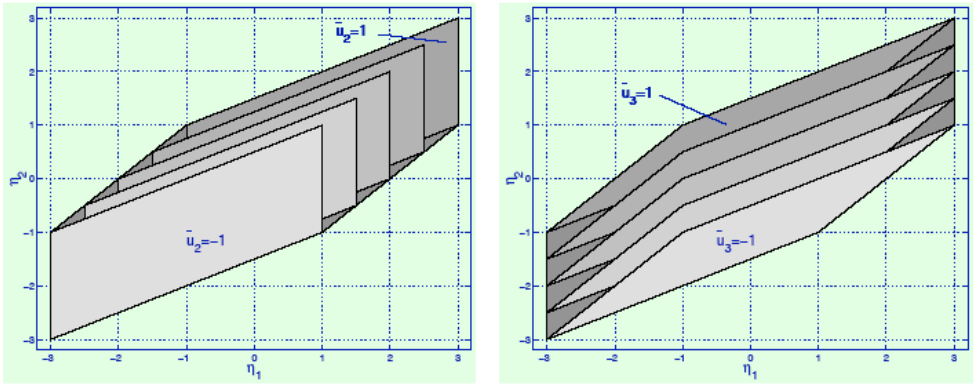

Fig. 15 Changes in the ADP set due to a hard-over failure control effector u_2 (left) u_3 (right).

D. ADP in the Case of Non-Affine Vehicle Dynamics

In this case, the Eqs. (1) and (2) are extended to include a nonlinearity that is non-affine in u, and the representation of the flight mode and uncertain parameters, resulting in the system of the form:

$$\dot{x}_1 = f_1(x_2) \tag{19}$$

$$\dot{x}_2 = f_2(x_1, x_2, u, \lambda, p) \tag{20}$$

where λ denotes the flight mode, and p is a vector of parameters whose values are not always known precisely.

The flight mode λ is a discrete variable taking values in $\{1,\cdots,M\}$. Here the flight mode refers to all operating conditions, including failure modes and battle damage, that change the flight dynamics of the vehicle. The parameter vector p may include variables such as moments of inertia and aerodynamic stability derivatives that may not be known precisely, for example, after battle damage. It is possible that the components of the vector p may be known with great accuracy in some flight modes, but in other modes they will need to be estimated in flight using input-output data.

Flight vehicles must also satisfy certain constraints. The constraints can be grouped into: (a) Path space constraints which limit kinematic variables: $g_1(x_1) \leq 0$, (b) Flight dynamic constraints which limit dynamic variables such as airspeed and accelerations: $g_2(x_1, x_2, \lambda, p) \leq 0$, and (c) Control input constraints which limit the control inputs u: $u \in S_u$. A control input is said to be feasible if it satisfies control input constraints. A trajectory is said to be

feasible if there is a feasible control input sequence which, when applied to the system, produces a state trajectory that satisfies both path space and flight dynamic constraints.

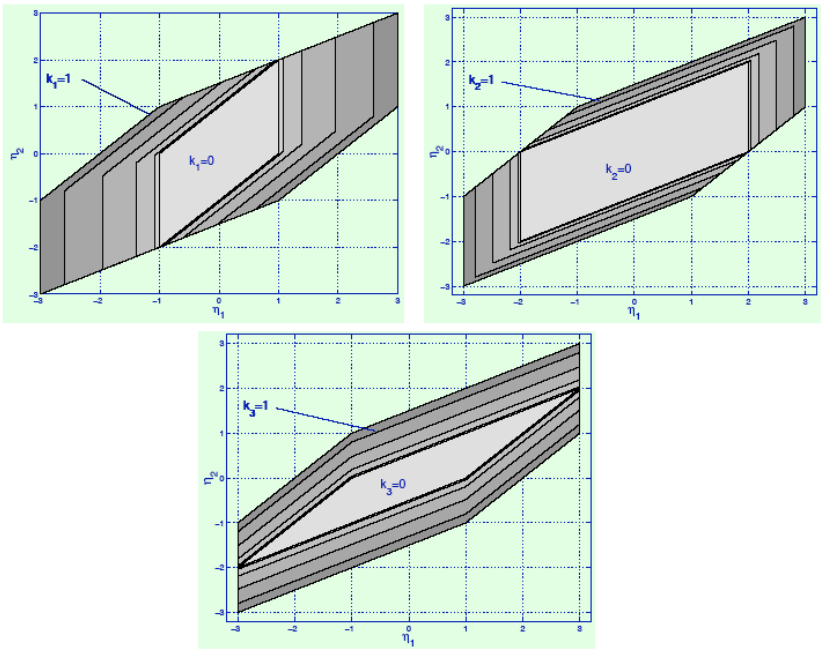

Fig. 16 Changes in the ADP set due to a loss-of-effectiveness failure of control effector u_1 (upper left), u_2 (upper right), and u_3 (lower figure).

Let the following variables be fixed: a state variable $x = [x_1', x_2']'$, a flight mode λ and a parameter p, leaving the control input u as the only variable in Eq. (2). Define the set:

$$\hat{S}(x, \lambda, p) = \{ f_2(x_1, x_2, u, \lambda, p) : u \in S_u(\lambda) \}$$

which is the set of all achievable rates of change of dynamic state variables at (x, λ, p). The ADP problem in the non-affine case is to characterize $\hat{S}$ for each x, λ, and p. Since the effect of u is additive in the affine case, the ADP problem can be stated without reference to x. If, in addition, $f(x) = Ax$, which corresponds to the linear case, the set $\hat{S}$ can be obtained by translating the set S_η by Ax. A consequence is that the shape of ADP set in the linear case is changed only by changes in λ and p. The nonlinear non-affine case is considerably different in these aspects. Though one of the objectives of ADP calculation is to characterize $\hat{S}$ exactly, its non-convexity causes computational difficulties. Hence some approximation is needed, and one possibility is to approximate $\hat{S}$ using a convex inner approximation. As an illustration, consider the region $\hat{S}$ shown on the left hand side in Figure 17. Choose a vector c_1 and define a hyperplane:

$$c_1' z = \gamma$$

where γ is a real number. The figure shows this plane, when $\gamma = b_1$, as cutting the set $\hat{S}$. If this plane is translated in a perpendicular direction as shown, it will eventually become tangent to $\hat{S}$ for some value of γ. Similarly, if one moves in the other direction, there will be another point (and another value of γ) at which tangency is achieved. These tangent points can be found by solving the following optimization problems:

$$\max_{z \in \hat{S}} c_1' z \quad \text{and} \quad \min_{z \in \hat{S}} c_1' z$$

i.e, at the tangent points $c_1' z$ is either a maximum or a minimum (geometrically, at these points, the set $\dot{S}$ lies on one side of the plane). By repeating this procedure for several vectors c_2, c_3,$\cdots$,c_k, the points that are on the boundary of the set to be approximated can be found. The polytope formed as the convex hull of these tangent points is a candidate inner approximation set. While it cannot be guaranteed that the procedure will produce an inner approximation as shown on the right hand side in Figure 17, this problem can be rectified by approximating $\hat{S}$ by a union of a finite number of polytopes instead by a single polytope.

VI. Verification and Validation of Intelligent and Adaptive Control Systems

Flight critical software and systems requirements assert that the occurrence of any failure condition that would prevent the continued safe flight and landing of the airplane shall be extremely improbable. These requirements are commonly specified in terms of a probability of loss of control (PLOC) due to failure being less than 10^{-7} for manned military aircraft, and 10^{-9} for unmanned aircraft. The PLOC requirement is currently verified through semi-exhaustive quantitative and qualitative test methods. However, in the case of autonomous aerial vehicles, one can expect an exponential growth in size and complexity of their flight critical systems and particularly their flight control software as a result of the need to perform advanced control and autonomous decision-making functions on-board (see Figure 18). The new required functionalities for the flight control system bring in unconventional architectures and algorithms, and software and hardware implementations. While traditional verification and validation (V&V) practices have produced safe and reliable software and systems, they will not be cost effective for autonomous systems that support these advanced control and decision-making capabilities. Affordable V&V of flight critical systems and software is by far the most important challenge facing both commercial and military aerospace industry in the United States. Advanced control and autonomous decision-making capabilities give rise to several major V&V issues. A list of some of these problems is included below, along with suggestions on how to address them.[40,41]

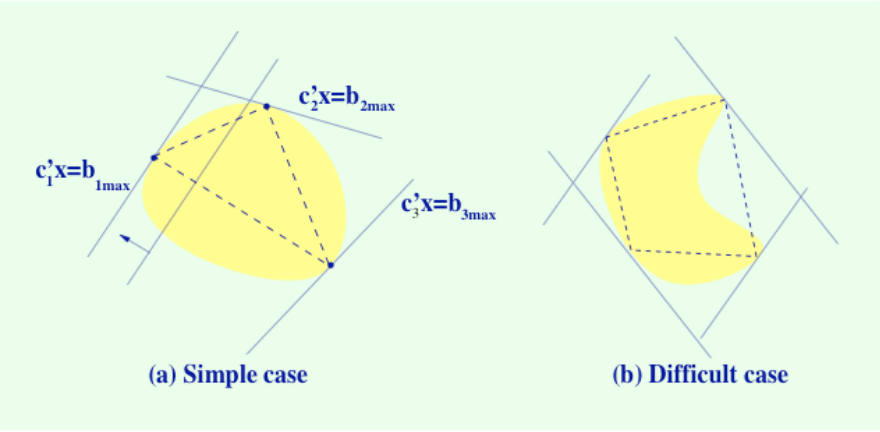

Fig. 17 Construction of a polytopic inner approximation.

A. Requirements Definition and Evaluation Criteria

Ideally, requirements and evaluation criteria are available in formal notation that is understood by a model checker, or in a manner that suggests a sufficient suite of tests. Several difficulties arise in practical applications. First, the requirements may be stated in natural language which must be interpreted/translated in a contextually correct manner. The translated requirement may not fully capture the intent of the original requirement leading to costly design iterations. Second, the requirements may be statistical in nature, for example, probability of loss of control (PLOC), and those regarding failure accommodation. Though precise, statistical requirements and evaluation criteria do not fit into the logical framework of existing formal verifiers. Third, for historical reasons, many aerial

vehicle requirements and evaluation criteria are in the frequency domain, for example MIL-STD requirements on short period damping, whereas the natural domain for analysis and design for nonlinear systems is time-domain. These and other difficulties indicate the need for a V&V-friendly requirements definition and performance evaluation standard.

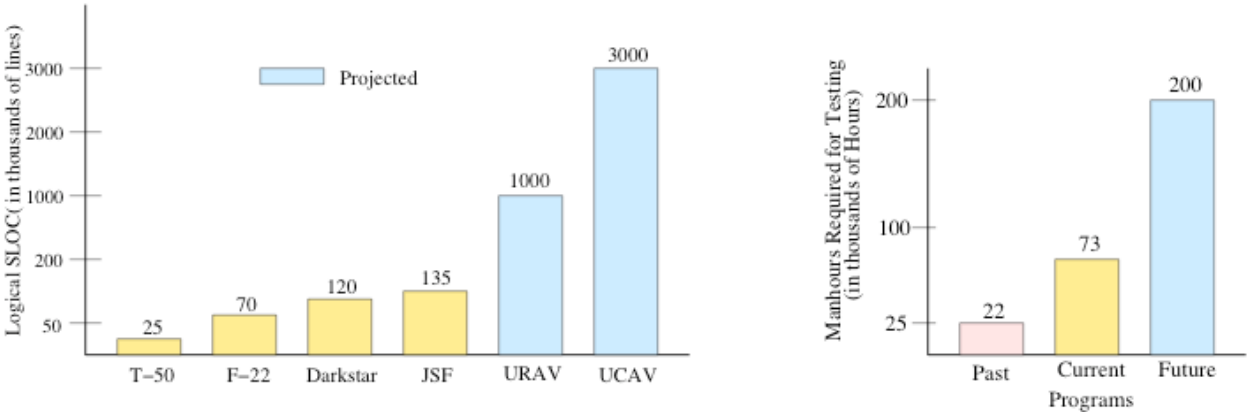

Fig. 18 Projected increases in source-line-of-code (SLOC) and testing for unmanned air vehicles[21] (URAV-unmanned reconnaissance air vehicle, UCAV-unmanned combat air vehicle).

Also, in current practice, past development effort is the primary guide for deriving component requirements from system-level requirements. This top-down approach may not be cost-effective or even feasible for intelligent autonomous systems which may exhibit new behaviors such as cooperation, coalition formation, and competition. Thus, a bottom-up approach of determining system-level behavior from component behaviors or perhaps a combination of both may be needed to verify and validate intelligent systems. These are deep current research issues in the area of hybrid and embedded systems.

B. Uncertainty and Adaptation

The most important feature that separates autonomous intelligent control systems from their traditional counterparts is uncertainty. While every control system can accommodate some amount of uncertainty, intelligent control systems are expected to perform in the presence of very large uncertainties by employing different adaptation and learning algorithms. However, on-line learning and adaptation raises the issue of their V&V, as discussed below.

By definition, the verification problem is to determine if a system satisfies a specification. A system can be thought of as a representation of all possible behaviors. In such a context, a specification describes a set of admissible behaviors, and a verified system behaves in an admissible manner. For example, in an intelligent FDI system, behavioral representations of known failure modes are used to detect and identify the failure, and safe behaviors are de fined as those in which the failure is accommodated. However, if there is a failure that falls outside a set of anticipated failure modes, the V&V problem quickly becomes intractable since both formal representation and related specifications are lacking in such a case. Hence the issue of how to represent the uncertainties and associated formal specifications are fundamental problems in the V&V of intelligent and adaptive control systems. It can be concluded that currently a formal V&V framework for adaptive and learning control systems does not exist, and that very little is known about efficient testing of such systems. An initial study discussing a possible framework for V&V of adaptive control systems is reported recently[40,41] and discussed below.

The closed loop system obtained from the hierarchical intelligent control system described in the previous sections, as well as a large class of existing flight control systems are hybrid systems with state-dependent switching of the form:

$$\dot{x} = f_{\lambda}(x) + g_{\lambda}(x)w \quad \text{if } x(t) \in X_{\lambda}$$

where $x(t) \in \mathrm{R}^{n_x}$ is the real-valued state, w is an exogenous input belonging to set W, λ is the discrete-valued state taking values in $\{1,2,\cdots,N\}$, and the subsets $\{X_1,\cdots,X_N\}$ form a partition of the state space R^{n_x}. The real-valued

state follows the dynamics given by f_i, g_i in the interior of the ith state space region X_i. When the state reaches the boundary of X_i, a formula (the mode switching logic) of the form:

$$h(x,\lambda,i) \geq 0$$

is evaluated to determine the next discrete-valued state. An approximate verification procedure for this class of systems has been recently developed.[40,41] The underlying computational approach is analogous to the behavioral approach to model checking of finite state systems.[25,29] In the behavioral approach to finite state systems, the system and the specification are represented by automata. The problem of checking if the system satisfies the specification then becomes a language emptiness problem. There are well-known combinatorial/graph theory algorithms to efficiently solve emptiness problem.[25] This approach for hybrid systems verification was chosen since it permits the use of linear matrix inequality (LMI)/convex programs for reach set computations and because many existing model checkers for finite state machines such as SPIN[29] use behavioral approach and can be adapted to this approach by equipping them with convex/LMI solvers. Even with the use of LMI/convex methods, most properties of hybrid systems cannot be verified exactly.[4] This is a consequence of the universal simulation properties of hybrid systems and Rice's theorem.[37] The above discussed verifier is approximate and iterative with the following guarantee: if the iterations terminate, then the system satisfies the requirements. Figure 19 shows the iterative process. Since engineers consciously put in safety margins during design, the iterations are expected to converge in practical applications.

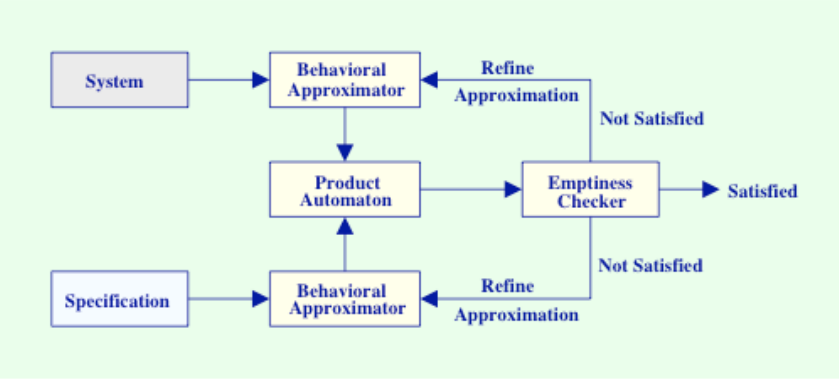

Fig. 19 Overall structure of the hybrid systems verifier.

C. Computational Complexity

The sheer size and complexity of intelligent control systems are beyond the capabilities of existing formal verifiers. Therefore, it is important to identify scalable algorithms and techniques. Abstraction, compositional reasoning, use of domain-specific methods and automatic test generation are some topics that should be investigated. These are currently topics of vigorous research in the formal methods area.[25,29–31,38] It is known that most properties of realistic intelligent systems cannot be formally verified. For example, a number of results on undecidability of stability properties of gain scheduled systems are currently available. These negative results have effected a compromise in the notion of solving a problem from the crisp yes/no to a probabilistic notion. Borrowing from the theory of learning,[44] the concept of a probably approximately correct (PAC) verifier is introduced. This verifier estimates, in polynomial-time, the probability that a certain property is satisfied by the intelligent system.* Another source of complexity comes from the fact that intelligent systems are open systems as opposed to closed systems whose behaviors are completely determined by the system state. An open system is a system that interacts with its environment and whose behavior depends on the interaction. It has been shown[33] that model checking for open systems is EXPTIME-complete for Computation Tree Logic (CTL) specifications. Here also a shift to probabilistic verification appears as an attractive direction for future research.

D. Translation/Model Extraction Tools

*A typical phenomenon in computation complexity is that, by considering probabilities rather than yes/no answers, efficient algorithms for hard problems can be designed. One of the primary reasons for the limited applicability of model checking and other formal tools is their insurance on a yes/no answer.

Engineers are less likely to use a formal verifier or an automatic test generator if it requires system representation in a format other than their native design/development language. For example, while control engineers understand and mainly work with SIMULINK/STATEFLOW diagrams, most model checkers require system representation in a different formal language (e.g. PROMELA for SPIN[29]). In order to gain acceptance of practicing engineers, software tools that can automatically translate control systems code from their native developmental language to the verification model need to be developed. This is an active research area at NASA (e.g. translation of Livingstone language to Symbolic Model Verifier[31]), Carnegie-Mellon University, and at other institutions. At present, except for the work reported in Refs. 24 and 41 there is no translation tool for hybrid systems making the verification process difficult at the early and mid-stages of flight vehicle development.

The growing use of software tools for design and simulation by aircraft manufacturers has also increased the prospects of extensive testing of control system designs before they are realized in software and implemented on the target processor since the detection and correction of errors in the early development phase are the most effective ways to reduce cost. Unfortunately, these software tools and translators are available only for a limited class of systems that does not contain intelligent and adaptive systems.

A typical phenomenon in computational complexity is that, by considering probabilities rather than yes/no answers, efficient algorithms for hard problems can be designed. One of the primary reasons for the limited applicability of model checking and other formal tools is their insistence on a yes/no answer.

VII. Conclusions

The design of fully autonomous intelligent flight control systems for both single and multiple UAVs is a formidable task. In this paper many of the issues that arise in the context of design of such complex controllers are described and some recent results are discussed.

In particular, a four-layer autonomous intelligent control architecture for UAVs is described, and related issues are discussed. The architecture consists of the following layers: (i) Redundancy Management Layer that consists of the on-line Failure Detection and Identification (FDI) and robust feedback Adaptive Reconfigurable Controller (ARC); (ii) Autonomous Trajectory Generation (ATG) layer whose role is to fit feasible trajectories through the desired way-points in real time; (iii) Autonomous Path Planning (APP) layer that generates way-points on-line in response to a dynamically changing environment; and (iv) Autonomous Decision Making (ADM) layer whose role is to assess the available control authority after failures, and make mission-related decisions in near-real time. Recent extensions of this architecture are also discussed and described in detail. At the end, a discussion is included on the V&V of intelligent and adaptive control systems and some recent results are presented.

The main contribution of the paper is an architecture for autonomous intelligent control of UAVs in which the layers are connected through the Achievable Dynamic Performance (ADP) calculation module. This results in a system in which all the decisions are made based on an optimum use of the current available resources.

Acknowledgments

This research was partially supported by the Defense Advanced Research Project Agency under the Contract No. DAAH01-00-C-R187 and NASA Dryden Flight Research Center under the Contract No. NAS4-02017 to Scientific Systems Company, Inc.

References

[1]Ahmed-Zaid, F., Ioannou, P., Gousman, K., and Rooney, R., "Accommodation of Failures in the F-16 Aircraft Using Adaptive Control," *IEEE Control Systems Magazine*, Vol. 11, No. 1, Jan. 1991, pp. 73-78.

[2]Beard, R. W., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., Mclain, T., and Goodrich, M., "Autonomous Vehicle Technologies for Small Fixed Wing UAVs," *Journal of Aerospace Computing, Information, and Communication*, (to appear).

[3]Bodson, M., and Groszkiewicz, J., "Multivariable Adaptive Algorithms for Reconfigurable Flight Control," *IEEE Transactions on Control Systems Technology*, Vol. 5, No. 2, March 1997, pp. 217-229.

[4]Blondel, V., and Tsitsiklis, J., "A Survey of Computational Complexity Results in Systems and Control," *Automatica*, Vol. 36, No. 9, 2000.

[5]Boeing Phantom Works, "Reconfigurable Systems for Tailless Fighter Aircraft -RESTORE (First Draft)," Contract No. F33615-96-C-3612, Scientific and Technical Reports, System Design Report, CDRL Sequence No. A007, St. Louis, MO, May 1998.

[6]Boskovic, J. D., Li, S.-M., and Mehra, R. K., "A Hybrid Fault-Tolerant Scheme for Flight Control Applications," *Proceedings of the 2000 AIAA Guidance, Navigation and Control (GNC) Conference*, Aug. 2001.

[7]Boskovic, J. D., Li, S.-M., and Mehra, R. K., "Robust Supervisory Fault-Tolerant Flight Control System," *Proceedings of the 2001 American Control Conference*, 25-27 June 2001.

[8] Boskovic, J. D., Bergstrom, S. E., Urnes, Sr., J. M., Mehra, R. K., Hood, M., and Lin, Y., "Performance Evaluation of an Integrated Retrofit Failure Detection, Identification and Reconfiguration (FDIR) System Using High-Fidelity and Piloted Simulations," presented at the *2004 SAE World Aviation Congress*, 2-4 Nov. 2004.

[9]Boskovic, J. D., Garagic, D., Byrne, J., Cosgrove, M., and Mehra, R. K., "Development of Intelligent Model Predictive Control Algorithms and a Software Design Toolbox for Autonomous Systems," Semi-Annual Report #7 for DARPA Phase II SBIR, Contract No. DAAH01-00-C-R187, July 2004.

[10]Boskovic, J. D., Chen, L., and Mehra, R. K., "Adaptive Control Design for a Class of Non-Affine Models Arising in Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, March–April 2004, pp. 209-217.

[11]Boskovic, J. D., and Mehra, R. K., "Robust Fault-Tolerant Control Design for Aircraft Under State-Dependent Disturbances," *Proceedings of the 2003 AIAA Guidance, Navigation and Control Conference*, 11-14 Aug. 2003.

[12]Boskovic, J. D., and Mehra, R. K., "A Multiple Model Adaptive Flight Control Scheme for Accommodation of Actuator Failures," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, July-Aug. 2002, pp. 712-724.

[13]Boskovic, J. D., and Mehra, R. K., "An Autonomous Hierarchical Control Architecture for Unmanned Aerial Vehicles," *Proceedings of the 2002 AIAA Guidance, Navigation and Control Conference*, 5-8 Aug. 2002.

[14] Boskovic, J. D., and Mehra, R. K., "A Decentralized Scheme for Autonomous Compensation of Multiple Simultaneous Flight-Critical Failures," *Proceedings of the 2002 AIAA Guidance, Navigation and Control Conference*, 5-8 Aug. 2002.

[15]Boskovic, J. D., and Mehra, R. K., "Failure Detection, Identification and Reconfiguration in Flight Control," *Fault Diagnosis and Fault Tolerance for Mechatronic Systems, Recent Advances Series*: *Springer Tracts in Advanced Robotics*, Vol. 1, edited by F. Caccavale and L. Villani, Springer Verlag, NY, 2002, Chap. 5.

[16]Boskovic, J. D., and Mehra, R. K., "Intelligent Adaptive Control of a Tailless Advanced Fighter Aircraft Under Wing Damage," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, Sept.-Oct. 2000, pp. 876-884.

[17] Boskovic, J. D., Li, S.-M., and Mehra, R. K., "Evaluation of the Properties of a Multiple-Model Reconfigurable Flight Controller on a 6 DOF Simulation," *Proceedings of the 2000 AIAA Guidance, Navigation and Control Conference*, Aug. 2000.

[18]Boskovic, J. D., and Mehra, R. K., Stable Multiple Model Adaptive Flight Control for Accommodation of a Large Class of Control Effector Failures", in Proceedings of the 1999 American Control Conference, San Diego, CA, June 1999.

[19]Brinker, J., and Wise, K., "Reconfigurable Flight Control of a Tailless Advanced Fighter Aircraft," *Proceedings of the 1998 AIAA Guidance, Navigation and Control Conference*, Vol. 1, 10-12 Aug. 1998, pp. 75-87.

[20]Bryson, A., and Ho, Y.-C., *Applied Optimal Control*, Hemisphere Publishing, 1975.

[21]Buffington, J., Crum, V., Krogh, B., Prasanth, R., Plaisted, C., Bose, P., and Johnson, T., "Verification and Validation of Intelligent and Adaptive Control Systems," AIAA Paper 2003-6603, April, 2003. See also AFRL's VVIACS program (Contact: Vincent.Crum@wpafb.af.mil or James.M.Buffington@lmco.com).

[22]Calise, A., Lee, S., and Sharma, M., "Direct Adaptive Reconfigurable Control of a Tailless Fighter Aircraft," *Proceedings of the 1998 AIAA Guidance, Navigation and Control Conference*, Vol. 1, pp. 88-97, 10-12 Aug. 1998.

[23]Chandler, P., Pachter, M., and Mears, M., "System Identification for Adaptive and Reconfigurable Control," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, May-June 1995, pp. 516-524.

[24]DARPA MOBIES Program, Goals and Challenges. Available online at http://dtsn.darpa.mil/ixo/programs.asp, 2000-2004 (cited Dec. 2004).

[25]Clarke, E., Grumberg, O., and Peled, D., *Model Checking*, MIT Press, Cambridge, MA, 2001.

[26]Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[27]Faiz, N., Agrawal, S. K., and Murray, R. M., "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities," *Proceedings of the 2000 AIAA Guidance, Navigation and Control Conference*, 14-17 Aug. 2000.

[28]Frazzoli, E., Daleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Paper No. AIAA-2000-4056, Aug. 2000.

[29]Holzmann, G., *PROMELA Manual*. Available online at http://spinroot.com/spin/Man/promela.html (cited Dec. 2004).

[30]Havelund, K., Lowry, M., and Penix, J., "Formal Analysis of a Spacecraft Controller Using SPIN," presented at the *4th International SPIN Workshop*, Nov. 1998.

[31]Pecheur, C., "The Livingstone Model-Based Diagnosis System." Available online at http://ase.arc.nasa.gov/ and http://ic-www.arc.nasa.gov/tech/index.php (cited Dec. 2004).

[33]Kupferman, O., Vardi, M., and Wolper, P., "Module Checking," TR98-302, Rice Univ. Technical Report, 1998.

[34]Latombe, J. C., *Robot Motion Planning*, Academic Publishers, Boston, MA, 1991.

[35]LaValle, S. M., and Kuffner, J. J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, May 2001, pp. 378-400.

[36]Narendra, K. S., and Balakrishnan, J., "Adaptive Control using Multiple Models," *IEEE Transactions on Automatic Control*, Vol. 42, No. 2, Feb. 1997, pp. 171-187.

[37]Papadimitriou, C., *Computational Complexity*, Addison-Wesley, 1994.

[38]Pecheur, C., "Verification and Validation of Autonomy Software at NASA," Preprint, 2002.

[39]Prasanth, R. K., Boskovic, J. D., and Mehra, R. K., "Computational Methods for the Verification of Adaptive Control Systems," SPIE Paper No: 5429-29, *Proceedings of the Signal Processing, Sensor Fusion, and Target Recognition XII*, *SPIE Defense and Security Symposium*, 12-14 April 2004.

[40]Prasanth, R. K., Boskovic, J. D., and Mehra, R. K., "Approximate Verification of a Class of Adaptive Control Systems," Proceedings of the American Control Conference, June 2004.

[41]Prasanth, R. K., Boskovic, J. D., and Mehra, R. K., "Assurance Technology for Software and Systems," DARPA Phase II SBIR Report, 2001-2004.

[42]Prasanth, R. K., Cabrera, J., Mehra, R., Purtell, R., and Smith, R., "Quality of Information in Autonomous Decision-Making," *Proceedings of the Second AIAA "Unmanned Unlimited" Systems, Technologies and Operations*, 2003.

[43]Prasanth, R. K., Boskovic, J. D., and Mehra, R. K., "Mixed Integer/LMI Programs for Low-Level Path Planning," *Proceedings of the 2002 American Control Conference*, 8-10 May 2002.

[44]Vidyasagar, M., *A Theory of Learning and Generalization with Applications to Neural Networks and Control Systems*, Springer, 1997.

[45]Scientific Systems Company, Inc., and Univ. of California Berkeley, "Autonomous Intelligent Health Monitoring for Robust Fault-Tolerant Control of Multi-Modal Systems," Final Report for NASA Phase I STTR, Contract No. NAS 2-03102, Jan. 2004.

[46]Wills, L., Sander, S., Kannan, S., Kahn, A., Prasad, J. V. R., and Schrage, D., "An Open Control Platform for Re-configurable, Distributed, Hierarchical Control Systems," *Proceedings of the Digital Avionics Systems Conference*, Oct. 2000.

[47]Yakimenko, O. A., "Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, Sept.-Oct. 2000, pp. 865-875.