MULTI-FUNCTIONAL AUTOPILOT DESIGN AND EXPERIMENTS FOR ROTORCRAFT-BASED UNMANNED AERIAL VEHICLES

David Hyunchul Shim[1], Hyoun Jin Kim[2], Hoam Chung[2], Shankar Sastry[3]

University of California, Berkeley, California

Abstract

In this paper, we present a hierarchical guidance and control system designed for rotorcraft-based unmanned aerial vehicles (RUAVs) for cooperative multi-agent scenarios. The issues of multi-agent system is resolved with a distributed hierarchical structure, which gradually transforms abstract mission commands into realtime control signals for multiple numbers of agents. The cooperative operations of multiple agents are realized by the centralized strategic planner, which interacts with other agents through high-bandwidth wireless communication system. The proposed design is implemented on a Berkeley UAV, *Ursa Magna 2,* and four other unmanned ground vehicles and validated thoroughly in a variety of tests from way-point navigation to pursuit-evasion games.

Introduction

There are numerous applications of unmanned vehicles in civilian or military operations where human participation is considered risky, unnecessary, and/or impossible. For these situations, Unmanned Aerial Vehicles (UAVs), typically based on fixed-wing airframe, have been successfully demonstrated their potentials in many applications. The Rotorcraft-based UAV (RUAV) has been considered as an attractive alternative because of their flight capabilities. The unique lift generation mechanism of rotorcrafts enables hover, vertical take off/landing, pirouette, and sideslip; these cannot be achieved by fixed-wing aircraft. These flight modes are often desired in many scenarios for reconnaissance, aerial surveying, and more. Remarkable progresses have been made in RUAV research during the last decade because of the progresses in key areas such as modeling, control theory, and small size electronics [3,4,5,9].

The Berkeley UAV research aims to synthesize, implement, and analyze a hybrid system consisting of multiple agents in cooperative scenarios. These agents actively operate, interact, and achieve the given abstract tasks using the provided autonomy and intelligence in a poorly known or completely unknown environment. This goal encompasses diverse fields of science and technology such as control theory, hybrid system theory, artificial intelligence, probabilistic reasoning, and vision-based servoing to name a few. In order to demonstrate these ideas, the implementation of UAV system is a crucial step. Therefore, in this paper, we present a novel approach for the synthesis of a flight control system (FCS) for UAVs based on a hierarchical hybrid system.

Figure 1 Berkeley RUAV, *Ursa Magna 2*, in an autonomous mission with Pioneer UGVs

[1] Postdoctoral Fellow, Electronics Research Laboratory, University of California, Berkeley
[2] Graduate Student, Department of Mechanical Engineering, University of California at Berkeley
[3] Professor, Department of Electrical Engineering and Computer Science, University of California at Berkeley
{hcshim, jin, hachung, sastry}@robotics.eecs.berkeley.edu

0-7803-7034-1/01/$10.00 ©2001 IEEE.

In the following sections, we will present 1) overview of flight control system (FCS) for UAVs, 2) hierarchical structure for UAV FCS, and 3) the application of the proposed FCS to three examples: pre-programmed waypoint navigation, dynamic waypoint navigation in a pursuit-evasion game, and high-speed target tracking control.

Flight Control System for UAVs

One of the most essential tasks of a UAV is to autonomously guide itself through the requested trajectories, or waypoints, in an autonomous manner. In order to achieve this goal, a UAV is equipped with onboard navigation sensors, real-time control units, and communication devices. For navigation sensors, the GPS-based INS is the most popular choice. The relatively poor accuracy of strap-down INS is nicely complemented by the use of high-accuracy GPS, which corrects the unbounded error of INS. Additional sensors such as ultrasonic sensors, laser range finders, barometric sensors are often deployed to acquire the environment-specific information such as relative altitude, i.e., the distance from the ground. The first two sensors are also able to detect the objects around the host vehicle, allowing collision detection/avoidance. For computer systems, a variety of CPUs, ranging from embedded microprocessors to general-purpose CPUs, are available for high-speed realtime applications. The computer systems typically run on realtime operating systems such as VxWorks™ or QNX™ to satisfy hard realtime requirements. For communication, or traditionally referred as telemetry, there are many choices of products varying in frequency, range, and protocol.

It is illuminative to examine the uniqueness of UAV FCS from conventional systems for manned vehicles, as presented in the following.

- **Autonomy**: The UAV FCS should be able to function with minimal supervision of human operators away from the vehicle. The autonomy, in diverse forms varying from simple *if-then* logic to sophisticated artificial intelligence, is the most distinctive feature of the UAV FCS. It is required to be aware of the current situation, make an optimal decision to achieve the goal in compromising situations, and communicate with the mission post or other agents to receive commands and share information.
- **Interface**: UAVs should accept incoming requests from human operators or other agents to achieve the given goal. When a human operator wants to send a command to a UAV, it actually goes through two interfaces: human-to-console interface and console-to-UAV FCS interface. The former interacts with human operator, receiving commands and displays the information downloaded from the UAV. It is often implemented with graphics user interface (GUI), which aims the maximal perception of the situation of UAVs. As the autonomy of UAV system improves, it is now of main interests to provide an efficient user interface to control multiple agents by a single operator. The console-to-UAV interface sends the human commands or computer-generated commands in a data structure and receives the UAV status. The data format has a tight relationship with the type of the communication channel. Typically, a UAV FCS accepts low- to high-level motion control commands, varying from simple motion commands to sophisticated behavioral commands. They allow external systems to guide the vehicle along the desired trajectory required for the mission.

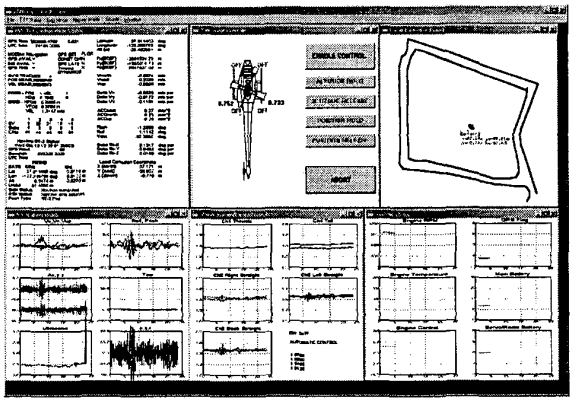


Figure 2 Graphical interface on ground station of Berkeley UAV testbed

- **Communication**: The role of communication in a FCS of UAV is more critical than an ordinary FCS in a manned vehicle for two reasons. First, the FCS in UAV should report the vehicle status and accept external commands in regular basis, typically at a faster rate than human voice communication. Second, with the support of high quality of service (QoS), a group of UAVs may perform missions in a tightly coordinated manner, which surpasses the bandwidth and accuracy of human decision-making and muscle control. When the FCS is combined with a wireless communication, a group of UAVs may function as a reconfigurable, multi-functional distributive system, which is far more flexible and robust than a single multi-purpose system.
- **Limiting factors**: in UAV applications, payload, space, and cost are often the major limiting factors as exemplified by the use of low-cost strapdown INS. Furthermore, UAVs often require more accurate navigation sensors because of the raised accuracy requirements for the accurate guidance of the host vehicle of much smaller size. Especially for smaller size of UAVs, the payload is the primary limiting factor in design and operation.

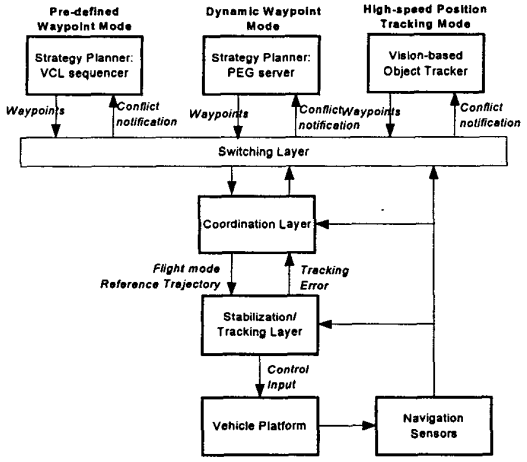


Figure 3 Multi-functional hierarchical structure of Berkeley RUAVs

Hierarchical Structure of FCS

In our research, we adopt a hierarchical structure in Figure 3 to implement a FCS that satisfies the attributes listed above. The hierarchical structure consists of strategic planners, a coordination layer, a stabilization/tracking layer, and a physical vehicle platform. The switching layer chooses the appropriate strategic planner for the given mission. The lower layers remain intact, preserving the integrity of the overall architecture. In the following, we introduce the role and the design process of each layer.

Vehicle Platform

Berkeley RUAVs are built on commercial off-the-shelf (COTS) radio-controlled helicopters. A number of helicopter platforms, varying in size and payload, have been adopted in our project. In this research, an industrial radio-controlled helicopter, Yamaha R-50, is used. The vehicle platform is integrated with onboard navigation computers and sensors for autonomous operation. A GPS-aided INS plays the central role for onboard navigation. Two ultrasonic sensors and four contact switches on the landing gear are also installed on the helicopter mainly for automatic take-off/landing. An optical engine RPM sensor regulates the engine at a constant speed in order to maintain the vehicle dynamic response close to the nominal operating point at which the dynamic model is acquired. The FCS software is implemented on QNX™ RTOS and responsible for sensor management, vehicle control, and communication. More detailed theoretical and practical issues considering in building an autonomous aerial vehicle are described in [12]. The implementation of Berkeley UAV/UGV testbed in Figure 4 shows that the proposed hierarchy is nicely preserved. The wireless communication plays the backbone of the information flow in this architecture.

Dynamic model identification

The acquisition of high fidelity system models of the target UAVs is a crucial step towards the successful design of high-performance flight control system. In general, however, it is often a challenging task due to its multi-input multi-output (MIMO), nonlinear characteristics, severe disturbance, and its wide flight envelope. The

helicopter dynamic model can be modeled utilizing a lumped parametric approach that considers a helicopter as a combination of main rotor, tail rotor, fuselage and stabilizer fins. We use linear, time-invariant parametric model valid in hover. The limited range of LTI model over operating points may be resolved by applying gain-scheduling method or more advanced Linear Parameter Varying (LPV) methods.

The RUAV dynamics is in general similar to that of full-size helicopters except that a RUAV typically shows significantly faster response due to the smaller inertia and faster rotor revolution. Therefore, a servorotor mechanism is augmented to the main rotor system in order to increase response time delay and damping. Since the servorotor dominates the main rotor dynamics, it should be properly accounted for in the template model. In this research, prediction-error method (PEM)[6] is applied to the collected data using the parametric model proposed by Mettler [2]. The identified system model is a six degree-of-freedom linear rigid body model with first-order servorotor dynamics [12,14].

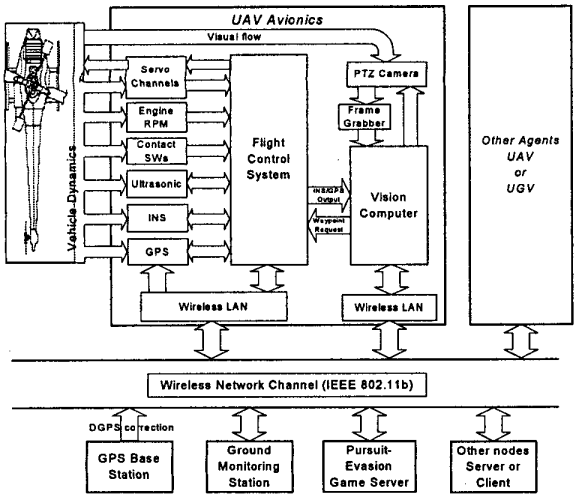


Figure 4 Scalable architecture for multi-agent scenarios

Stabilization/Tracking Layer

The unstable RUAV dynamics needs proper stabilization using feedback control, which is performed by the onboard realtime controller. The controller can be designed either using classical control theory or MIMO state-space control theories. The favored method by industry or military community is dominantly the classical SISO approaches due to the simple and intuitive nature. In our research, a multi-loop SISO controller is used [14]. It is briefly mentioned that, parallel with this conventional approach, a linear robust control system designed with μ -synthesis theory has been successfully applied to Berkeley UAV [12].

As presented in [14], the flight controller demonstrated a stable response over two minutes with ±0.5m accuracy in x and y direction. Roll, pitch, translational velocity in x and y directions are regulated very well altogether. The altitude regulation shows outstanding performance of ±0.1m error and the heading is also regulated within ±3 degrees.

Coordination Layer

A mission of a UAV typically consists of take-off, a number of waypoints, and landing. The interim flight patterns are further decomposed into the sequence of flight modes such as hover, pirouette, longitudinal/lateral flight, cruise, ascent/descent, and turn, as depicted in Figure 5. The coordination layer is responsible for triggering the proper control law of the stabilization/tracking layer to execute each of these flight modes in a preprogrammed sequence or dynamically upon request. The coordination layer in charge of the waypoint navigation is between the stabilization/ tracking layer and the strategic planner.

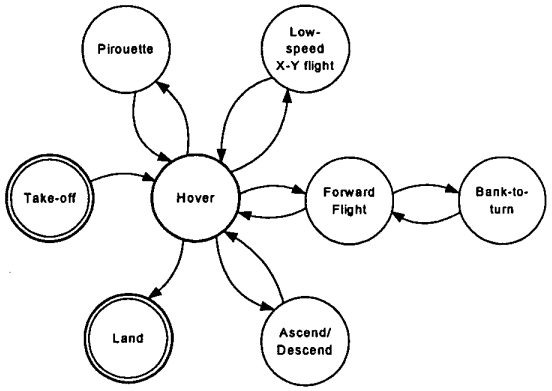


Figure 5 Finite-state transition diagram of autonomous helicopter flights

In designing such a waypoint navigator, we introduce a novel framework, the Vehicle Control Language, or *VCL*. VCL is a script language that specifies the given mission with the provided command set in Figure 6. Via VCL, we provide a layer for the isolation and abstraction between the strategic planner and the stabilization layer. A VCL module consists of the user interface part on the ground station, the language interpreter, and the sequencer on the UAV FCS. When a flight pattern for a UAV is given, the VCL code may be generated using a graphic user interface, or hand-written. The generated VCL is stored in a ASCII file and uploaded to the flight computer, which executes the VCL in a sequential manner.

TakeoffTo <coord>{**abs,rel**} : perform autonomous take-off to certain target point

Hover <coord>{**abs,rel**}
{**heading=**<heading>{**deg,rad**}}
duration>{**sec,min**}
: hover with given heading angle for given time

FlyTo <coord>{**abs,rel**} {**vel**
<velocity>{**mps,kmps,fps,knots,mph**}}
{**passby,stopover**} {**autoheading,**
heading=<heading>{**deg,rad**}}
: cruise to certain waypoint stopping over or passing by

MoveTo <coord>{**abs,rel**} {**vel**
<velocity>{**mps,kmps,fps,knots,mph**}}
{**autoheading, heading=**<heading>
{**deg,rad**}}
: move to certain way point to stopover with fixed heading

BankToTurn <heading change>{**deg,rad**}
{{**radius**}<radius>{**m,ft**}}
{{**vel**} <velocity>
{**mps,kmps,fps,knots,mph**}
: perform bank-to-turn during cruise

Land : command the vehicle to land

Figure 6 Vehicle Control Language Syntax

Strategic Planner

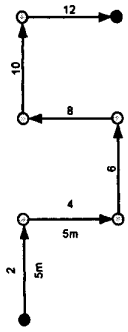
On the top level of the hierarchy in Figure 3, there lie strategic planners, which are made specifically for given target scenarios. As now, three strategic planners have been implemented: batch VCL mode, pursuit-evasion game server using dynamic VCL, and vision-based ground object tracking. These modules reside either in onboard flight control system or in ground stations, depending on the system configuration. It is called *centralized* if one strategic planner supervises all of the subordinate layers. If the strategic planners are running simultaneously in multiple agents by sharing information, it is called *decentralized* or *distributed.* The latter is of particular interests these days because of their advantages in robustness and flexibility, and the challenges in implementation, especially in terms of synchronization and communication. In our research, we choose the centralized approach as the initial step to avoid the implementation issues of decentralized systems. In the future, however, we aim to achieve a fully decentralized hierarchical system.

Experiments

In this section, we evaluate the effectiveness of the proposed hierarchical FCS in a series of test flights of three distinct scenarios. Three examples are: 1) batch (or preprogrammed) VCL mode, 2) dynamic VCL mode during a pursuit-evasion game (PEG), and 3) high-speed position tracking assisted by the onboard vision computer.

Batch VCL mode

In this mode, the VCL execution module assumes the highest hierarchy on the guidance of the test UAV, Ursa Magna 2. A lawn-mowing pattern as shown in Figure 7 is used as a benchmark trajectory. The VCL codes are generated manually and uploaded to the FCS as a text file. The flight mode, waypoint, and other optional parameters are extracted in each line of VCL and then sent to the coordination layer. Upon the reception of new VCL command, it activates a suitable control module for the current flight mode associated with the target waypoint and other options. The stabilization/ tracking layer generates real-time control output at 50Hz for the actuators on the host UAV. The navigation measurements are fed into all the layers for feedback control and other supervisory tasks.




```
Hover (0,0,0)rel heading=270deg duration=10sec;
FlyTo (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover (0,0,0)rel heading=0deg duration=10sec;
FlyTo (5,0,0)rel vel=0.5mps stopover autoheading;
Hover (0,0,0)rel heading=270deg duration=10sec;
FlyTo (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover (0,0,0)rel heading=180deg duration=10sec;
FlyTo (-5,0,0)rel vel=0.5mps stopover autoheading;
Hover (0,0,0)rel heading=270deg duration=10sec;
FlyTo (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover (0,0,0)rel heading=0deg duration=10sec;
FlyTo (5,0,0)rel vel=0.5mps stopover autoheading;
Hover (0,0,0)rel heading=270deg duration=10sec;
```

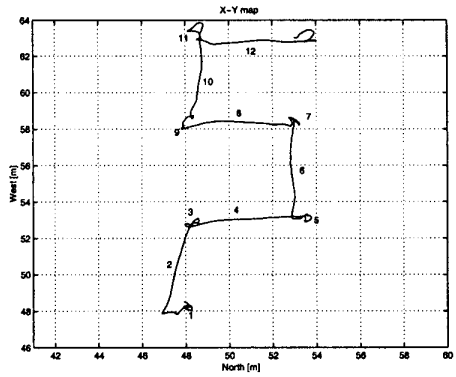


Figure 7 A VCL code for lawn-mowing pattern and flight experiment result

Dynamic VCL mode: PEG example

In this part, we evaluate the performance of dynamic VCL in an interesting scenario: the pursuit-evasion game [13]. The PEG is a game scenario, whose goal is to "catch[1]", in a finite time, evaders in a given field with pursuers, which may be commanded by a number of pursuit algorithms. The initial locations of evaders are unknown *a priori.* The pursuers build probabilistic maps of the possible locations of evaders using their sensing devices, which are typically vision-based. In this scenario, the group of pursuers, consisting of UAVs and/or UGVs, is required to go to the requested waypoints, take measurements of the location of themselves and of any evaders within its detection range, and report the measurement as well as their current locations to the PEG strategic planner. This measurement is used to compute the waypoint of pursuers at next time frame and sent to the pursuers in a VCL form via wireless communication. The onboard VCL execution module in the coordination layer processes the incoming VCL commands in a similar manner as the batch VCL case. In the experiment setup, the PEG algorithm is implemented in MATLAB/Simulink, which is modified to run in realtime using blocking socket of TCP/IP communication [13].

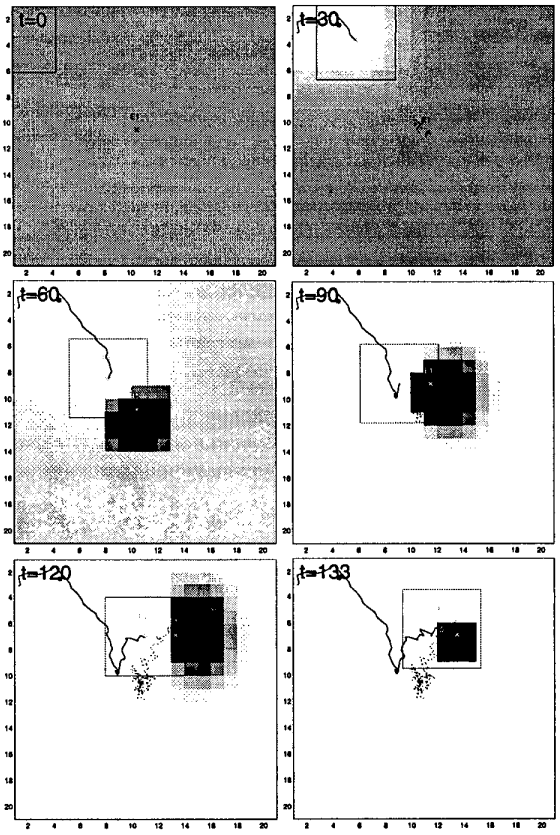


Figure 8 Snapshots of 1 vs. 1 Pursuit-Evasion Game (P: Pursuer UAV, E: Evader UGV)

In Figure 8, a result from a PEG game, one aerial pursuer vs. one ground evader, is shown. It is noted that the number of participating pursuers and evaders can be easily changed by simply adding or deleting Simulink blocks for TCP/IP communication. We chose the setup of one aerial pursuer so that the load of UAV is maximized. When the game starts, the UAV initially remains hover until it receives the starting sign from the PEG server. The ground robot roams in the given field of 20m x 20m. The graph shows two trajectories: one for the pursuer UAV and the other

[1] With reality constraints, an evader is considered as caught when it is approached by a pursuer within a certain range (e.g., 1.5m) and it is in the pursuer's detection region.

for the evader UGV. The snapshots in Figure 8 show the progress of PEG. Along with the trajectories, it also shows the probabilistic map shown as the gray-scale background and the visibility region denoted as a square. The UAV pursuer catches the evader in 133 seconds. This experiment shows that the dynamic VCL performs well in a hierarchical structure for multi-agent scenarios such as the pursuit-evasion game.

High-speed position tracking

In this scenario, we consider the situation when a UAV is required to track a moving ground object, whose location is measured and transmitted by an external source such as a vision computer. In this setup, a specially tuned waypoint navigator is activated to process the high-rate position request, at 3 Hz in this case.

The vision computer estimates the location of the ground target using a detection algorithm utilizing a special marker [15]. In Figure 9, the trajectories of UAV and UGV are shown. The FCS shows satisfactory tracking performance with small error due to wind gust. In the middle of the experiment, it is noticed that the vision computer ceased sending the reference trajectory about 8 seconds. The UAV FCS demonstrated its fail-safe feature against this adversary fault by standing by in hover mode until a new waypoint command is received.

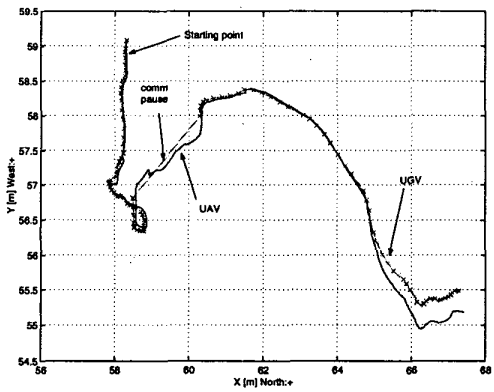


Figure 9 Tracking control of ground target

Conclusion

In this paper, we have shown the effectiveness of a hierarchical structure for a flight control system of a UAV system. By introducing a switching layer into the hierarchy, the flight control system could perform various tasks by the supervision of corresponding strategic planners. Three examples are employed for the evaluation of the proposed architecture. The experiment results prove that the multi-functional FCS for Berkeley UAV shows satisfactory performances in the all three cases. Especially in the PEG scenario, it is shown that the proposed hierarchical FCS seamlessly accomplishes the multi-agent scenarios. Further research effort will be exercised to expand the capability of the FCS with increased robustness.

Acknowledgement

The authors would like to thank the BEAR team, especially Rene Vidal, Omid Shakernia, and Cory Sharp for their collaborations on PEG and position tracking research. This research was supported by the ONR grants N00014-97-1-0946 and N00014-00-1-0621, and ARO MURI grant DAAH04-96-1-0341.

References

[1] R.W. Prouty, *Helicopter Performance, Stability and Control*, Krieger Publishing Company, 1995.

[2] B. Mettler, M. B. Tischler, T. Kanade, "System Identification of Small-Size Unmanned Helicopter Dynamics," *American Helicopter Society 55th Forum*, Montreal, Quebec, Canada, May 1999.

[3] H. Shim, T. J. Koo, F. Hoffmann, S. Sastry, "A Comprehensive Study of Control Design for an Autonomous Helicopter," *37th IEEE Conference on Decision and Control*, pp. 3653-3658, 1998.

[4] C. P. Sanders, P. A. DeBitetto, E. Feron, H. F. Vuong, N. Leveson, "Hierarchical Control of Small Autonomous Helicopters," *37th IEEE Conference on Decision and Control*, pp. 3629-3634, 1998.

[5] J. E. Corban, A. J. Calise, J. V. R. Prasad, "Implementation of Adaptive Nonlinear Control for Flight Test on an Unmanned Helicopter," *37th IEEE Conference on Decision and Control*, pp. 3641-3646, 1998.

[6] L. J. Ljung, *Matlab System Identification Toolbox User's Guide* , The Math Works, Inc.,1997.

[7] Anon., Rotorcraft System Identification, AGARD Advisory Report, 1991.

[8] D. J. Walker and I. Postlethwaite, "Advanced Helicopter Flight Control using Two-Degree-of-Freedom H_∞ Optimization," Journal of Guidance, Control, and Dynamics, Vol. 19, No. 2, pp. 461-468, March-April 1996.

[9] P. Bendotti, and J. C. Morris, "Robust Control for a Model Helicopter," *Proceedings of the American Control Conference,* pp. 682-687, Seattle, Washington, June 1995.

[10] J. N. Rozak, and A. Ray, "Robust Multivariable Control of Rotorcraft in Forward Flight," *Journal of the American Helicopter Society,* pp. 149-160, April, 1997.

[11] G. J. Balas, J. C. Doyle, K. Glover, A. Packard and R. Smith, *μ-Analysis and Synthesis Toolbox*, The MathWorks, Inc., 1995.

[12] D. H. Shim, "Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles", *Ph.D. Dissertation,* University of California, Berkeley, 2000.

[13] R. Vidal, O. Shakernia, H. J. Kim, H. Shim, S. Sastry, "Multi-Agent Probabilistic Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles," submitted to *IEEE Transactions on Robotics and Automation*

[14] D. H. Shim, H. J. Kim, S. Sastry, "Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles", *AIAA Guidance, Navigation and Control Conference,* Denver, August 2000

[15] C. S. Sharp, O. Shakernia, S. S. Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle," *IEEE International Conference on Robotics and Automation,* pp. 1720-1727, Seoul, Korea, 2001.