AUTONOMOUS FLIGHT CONTROL AND HARDWARE-IN-THE-LOOP SIMULATOR FOR A SMALL HELICOPTER

T. L. Ng * **P. Krishnamurthy** * **F. Khorrami** *
S. Fujikawa **

* *Control/Robotics Research Laboratory(CRRL)*
Dept. of Elec. and Comp. Engg., Polytechnic University
Six Metrotech Center, Brooklyn, New York 11201.
** *IntelliTech Microsystems, Inc., 17001 Science Dr.*
Suite 119, Bowie, MD 20715.

Abstract: The paper describes hardware development for a small helicopter and a hardware-in-the-loop (HITL) simulator for flight control system and obstacle avoidance testing and validation. The proposed obstacle avoidance system (OAS) is briefly outlined. The HITL simulator is developed to closely mimic the behavior of the actual flight hardware so as to easily develop, debug, and test the flight control system on the ground thereby minimizing the mission risk. The helicopter in use is a small X-Cell gas turbine retrofitted to achieve high bandwidth control of the helicopter with the ultimate goal of achieving obstacle avoidance once the inner-loop flight control system is tested. The sensor utilized for rigid body attitude and position determination is a navigation-quality Inertial Mesurement Unit (IMU). A specialized dual processor board has been designed and developed for implementation of the control system and the obstacle avoidance algorithm. Furthermore, the HITL simulator includes the actual helicopter itself for purpose of driving the servos on the helicopter and a board for mimicking the IMU output. Preliminary control system designs and testing on the HITL simulator and the path-planning and obstacle avoidance algorithm (GODZILA) are presented in this paper. Copyright© 2005 IFAC

Keywords: Autonomous vehicles, helicopter control, path planning, obstacle avoidance, helicopter dynamics.

1. INTRODUCTION

There are many civilian (e.g., weather forecasting, traffic monitoring, police operation, fire fighting) and military applications (e.g. surveillance, target identification, ordinance delivery, communication relay) for the use of unamanned aerial vehicles (UAV). Furthermore, in certain applications, unavailability of a runway and requirement of tight turn radius limit the use of fixed wing aircrafts. Hence, autonomous rotary wing aircrafts continue to have an important role in unmanned systems. Current FAA requirements analyses identify two possible means by which UAVs may be accepted into civilian airspace: 1) requiring all UAVs to have the necessary transponder hardware to identify themselves to the control tower or 2) requiring UAVs to have autonomous obstacle avoidance systems (OAS). This latter approach of requiring OAS on UAVs is favored by the industry. An autonomous unmanned aircraft equipped with an

[1] This work is supported in part by the ARO under contract no. W911NF-04-C-002 and by IntelliTech Microsystems, Inc.
[2] Author Emails: tlng@crrl.poly.edu, pk@crrl.poly.edu, khorrami@smart.poly.edu, sfujikawa@imicro.biz

OAS will be able to carry out a multitude of missions requiring flight to a designated target without benefit of remote piloting or any prior knowledge of the local geography. In this paper, we describe our on-going efforts on OAS and development of a small autonomous helicopter with ultimate objective of having an OAS implemented on the vehicle.

There has been much interest in UAVs in the past few decades; however, miniaturization of sensors, electronics, and fast microprocessors in the past decade have improved feasibility of small autonomous vehicles (Bendotti and Morris) and (Johnson and Kannan). Several papers on application of various nonlinear control techniques to helicopter dynamics have been reported. The approaches include differential geometric approaches (Isidori *et al.*), approximate feedback linearization (Koo and Sastry), backstepping based design (Frazzoli *et al.*), forwarding-based techniques (Mazenc *et al.*), linear robust multi-variable control (Yue and Postlethwaite,Civita *et al.*), nonlinear H_∞ control (Kung *et al.*), fuzzy logic control (Shim *et al.*), and control laws based on mimicking human pilots (Gavrilets *et al.*).

The objective of the current effort is to develop an autonomous helicopter for purpose of implementation of our proposed OAS (GODZILA) in unknown environments. Due to lack of space, GODZILA will be briefly outlined in this paper and some numerical simulations will be provided to show its efficacy. The important point to emphasize is that GODZILA was developed for the purpose of obstacle avoidance on small to miniature vehicles for which space, weight, power, and computational resources are at a premium. For this reason, GODZILA is a low resource algorithm shown to be effective in many simulation studies. This paper outlines the development of our small autonomous helicopter and the OAS. Specifically, we will focus on the Hardware-In-The-Loop simulator. The helicopter dynamics and its real-time simulation are described in Section 2. Sections 3 and 4 contain descriptions of the experimental testbed and the HITL simulator, respectively. The inner-loop control design and its validation on the HITL simulator are provided in Section 5. The GODZILA path-planning and obstacle avoidance algorithm is described in Section 6.

2. HELICOPTER DYNAMICS AND ITS REAL-TIME SIMULATION

As described in (Heffley and Mnich), a minimum-complexity mathematical model of the helicopter dynamics consists of fourteen states: the twelve rigid-body states and two states for the tip-path-plane orientation (flapping angles). Let $p = [p_t^T, p_r^T]^T$ where $p_t = [x, y, z]^T$ represents the Cartesian coordinates of the origin of the body-fixed frame as measured in the inertial frame and $p_r = [\theta_x, \theta_y, \theta_z]^T$ specifies the Euler rotation angles. Denote the rotation matrix transforming vectors from the body-fixed frame to the inertial frame by R_i^b. The body-fixed frame is constructed with X-axis pointed towards the helicopter nose, Y-axis towards the left, and the Z-axis upwards. Let the translational and angular velocities of the body relative to the inertial frame and expressed in the body-fixed frame be denoted by v_t and v_r, respectively, and let $v = [v_t^T, v_r^T]^T$. Then, the rigid-body kinematics and dynamics of the helicopter are given by

$$\begin{aligned} \dot{p} &= J(p)v \\ M_{RB}\dot{v} + C_{RB}(v)v &= \overline{F} \end{aligned} \tag{1}$$

where M_{RB} and $C_{RB}(v)$ are the 6×6 inertia and Coriolis matrices, respectively, $\overline{F} = [F^T, \tau^T]$ is the generalized force vector with F being the net force and τ the net torque on the helicopter, and $J(p)$ is the position Jacobian. The matrices M_{RB}, $C_{RB}(v)$ and $J(p)$ are given by

$$M_{RB}=\begin{bmatrix} mI_{3\times3} & -mS(p_{G,b}) \\ mS(p_{G,b}) & I_b \end{bmatrix} \tag{2}$$

$$C_{RB}(v)=\begin{bmatrix} mS(v_r) & -mS(v_r)S(p_{G,b}) \\ mS(p_{G,b})S(v_r) & -S(I_b v_r) \end{bmatrix} \tag{3}$$

$$J(p)=\begin{bmatrix} R_i^b & 0_{3\times3} \\ 0_{3\times3} & J_r(p_r) \end{bmatrix} \tag{4}$$

$$J_r(p_r)=\begin{bmatrix} 1 & s_x t_y & c_x t_y \\ 0 & c_x & -s_x \\ 0 & s_x/c_y & c_x/c_y \end{bmatrix} \tag{5}$$

where $p_{G,b}$ denotes the coordinates of the center of gravity of the helicopter in the body-fixed frame, m is the mass, I_b is the inertia matrix, $S(a)$ denotes the skew-symmetric matrix function of vector a, and $s_x = \sin(\theta_x)$, etc.

The generalized force vector $\overline{F}$ consists of a component due to gravity and components due to aerodynamic interaction with the helicopter parts. Considering a lumped model of the helicopter, the helicopter force and torque generating parts are the main rotor, tail rotor, horizontal stabilizer, vertical stabilizer, and fuselage. In hover or low-velocity forward flight, the generalized force vector components due to the horizontal stabilizer, vertical stabilizer, and fuselage are negligible. Hence, $\overline{F}$ is given by

$$\begin{aligned} \overline{F} &= (R_i^b)^T[0, 0, -g, 0, 0, 0]^T \\ &\quad +[F_{mr}, \tau_{mr}]^T + [F_{tr}, \tau_{tr}]^T \end{aligned} \tag{6}$$

where g is the acceleration due to gravity, F_{mr} and τ_{mr} are the force and torque due to the main rotor, and F_{tr} and τ_{tr} are the force and torque generated by the tail rotor.

The main and tail rotor models are based on the autogyro theory presented in (Heffley and Mnich). Thrust and induced velocity are computed assuming a uniform-flow distribution. The tip-path-plane orientation (longitudinal and lateral flapping angles a_1 and b_1) is modeled as two first-order lags so that the main rotor appears as a force actuator with a lag. The dihedral effect is included through the variables $\frac{da_1}{dv_{t,1}}$ and $\frac{db_1}{dv_{t,2}}$ which appear in the first-order flapping equations. The modeling of the tail rotor is similar to that of the main rotor except that flapping degrees of freedom are not included. The effects of the main rotor collective, the tail rotor collective, the pitch cyclic, the roll cyclic, and the throttle are included in the force and torque expressions.

The aforementioned helicopter dynamics are implemented and optimized to run in real-time on a PC with a graphical display showing the helicopter position and attitude and telemetry data.

3. HARDWARE DESCRIPTION

The overall hardware utilized for the autonomous flight has the following components: (a) X-Cell gas turbine helicopter, (b) an in-house developed dual-processor with FPGA board to interface to all sensors, actuators, and other devices on board and to be utilized as the auto-pilot, (c) a power distribution board and batteries, (d) a navigation quality Inertial Measurement Unit (IMU), (e) a wireless transceiver with a range of 60 miles, (f) a miniature camera and a wireless transmitter for video transmission, (g) a GPS, and (h) a ground station.

3.1 X-Cell Gas Turbine Helicopter

An X-Cell gas turbine helicopter has been modified specifically to function as a UAV (Figure 1). Modifications include an extended tailboom for additional stability, specially built composite main rotor blades, replacement of plastic parts with composite or aluminum parts in critical areas, relocation of control servos for optimum power output and control response, utilization of a gasoline powered ignition engine, and addition of an engine/rotor governor. The combination of the specialized rotors and the powerful engine allows the helicopter to carry payloads in excess of 10 pounds. The servos used in the helicopter are high-torque, high performance servos designed for UAV applications. The landing gear is also extended so that the IMU and other needed hardware can be mounted under the helicopter.

3.2 Dual-Processor with FPGA Auto-Pilot Board

To satisfy the computational and processing requirements on-board the small helicopter, we developed a two-processor hardware architecture for the auto-pilot. One processor will be utilized for inner-loop control system implementation whereas the second processor will be utilized for the Obstacle Avoidance System (OAS) computations and sensor processing. Since there

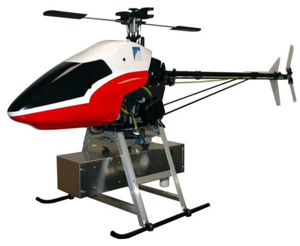


Fig. 1. X-cell helicopter.

are a large number of sensors to be processed with the IMU itself updating at a high rate, we have further included a Field Programmable Gate Array (FPGA) chip on board to facilitate the I/O processing and protocol handling for various sensors. The FPGA also acts as the communication interface between the two processors. This architecture provides the needed flexibility for further extending the hardware. The board with one RCM3200 processor plugged into it is shown in Figure 2. The second processor would be plugged into the back of the board at the same position.

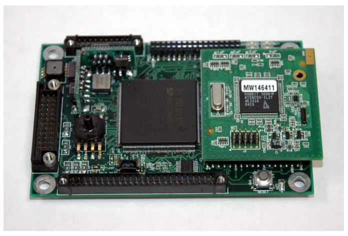


Fig. 2. Autopilot board.

This board additionally includes hardware solutions for communication and interfacing, onboard expansion slots for other instrumentation and daughter boards, ten PWM output channels (programmable resolution and pulse period), variety of serial interfaces (e.g., RS232, RS485, SDLC, etc.), four optical encoder inputs, eight RC input channels, safety features for auto-pilot and flexible hand-off to RC mode on a CPLD chip, and features for interfacing, programming, and monitoring from a PC using serial interface.

To interface with the IMU and the RCM3200, the FPGA incorporates the serial SDLC interface used by the IMU along with CRC and checksum error detection, a dual port RAM to exchange data with the RCM3200, 12-bit PWM generators to drive the servos, and all necessary interfaces between the two processors and the other hardware on board. The hand-off and safe-mode mechanisms for radio link or hardware failure are handled by a separate CPLD chip on the autopilot board for added safety.

3.3 Other Components

The main sensor for providing position and attitude of the vehicle is a navigation quality IMU. Due to the low drift rate on the IMU, the output is integrated using quaternions to generate the vehicle rigid-body state vector without the need for a GPS. The IMU is accurate enough to fly the vehicle for tens of seconds without GPS update.

A power distibution board has been designed to provide the various voltage levels (e.g., 3.3 V, 5 V, ± 15 V, etc.) required by various sensors. Voltage level monitoring has also been included on this board to alert the ground station of low power situation. We have also included other payloads such as wireless data and video modems/transceivers.

A ground station has also been developed to collect all the telemetry data and to develop a user interface between the ground crew and the helicopter.

4. HARDWARE-IN-THE-LOOP SIMULATOR

It is imperative that flight control system be validated on the ground before actual flight testing to reduce hardware risk and to test and validate the integrated software and hardware. To this end, a HITL simulator has been developed for our overall system to mimic the actual flying helicopter and its systems as closely as possible. Our HITL simulator has the following components:

(1) A real-time software simulation of the helicopter dynamics. This model is based on the minimum model complexity helicopter dynamics outlined earlier in the paper. This model is of modest complexity that faithfully simulates the main effects of helicopter dynamics but at the same time is amenable to control design and real-time implementation.
(2) The actual dual-processor auto-pilot board. This board would contain the exact flight control system that will be implemented on the helicopter. This will provide a high degree of confidence that the flight control system has been debugged and performs as expected. The controller is implemented at 50Hz.
(3) Generating the IMU data that would be required by the auto-pilot. The IMU outputs an SDLC serial format (with a high clock frequency). This needs to be generated in hardware as an input to the auto-pilot as if the actual IMU was providing real-time data to the auto-pilot during the flight.
(4) Including the servo dynamics on the helicopter and providing the servo positions to the software simulator. To include the actual servo dynamics in the HITL simulator, we have tapped into the potentiometers on the servos on-board the helicopter to measure servo positions. These analog signals are read by a multi-channel A/D and the serial output is sent to the RCM3200 serial port on board the IMU simulator to be transmitted to the PC running the Fortran code via the serial interface to the PC. This essentially closes the loop and allows the actual helicopter servos to be included in the HITL simulator.
(5) Interface and synchronization mechanisms among the software simulator, the IMU simulator, the auto-pilot, and all the component hardware. The utilized scheme results in a closed-loop delay of one sample (20ms).

4.1 IMU Simulator Hardware

To generate the actual SDLC serial format that IMU generates and the auto-pilot board expects as an input, we developed a hardware board (called IMU simulator) that would interface with the PC running the Fortran simulator code. The IMU simulator board is essentially the same board as the dual-processor board with the exception that the FPGA hardware is redesigned to do the inverse process as the FPGA board on the auto-pilot. That is the FPGA on the IMU simulator would receive the six incremental linear velocities and angular positions and would generate an SDLC signal with proper open and start flag, CRC error detection, Checksum error detection, and proper coding. This board communicates with the PC running the real-time helicopter dynamic simulator via a serial interface sending data to the RCM3200 processor on the IMU simulator. Due to graphics and real-time implementation, the real-time simulator runs at 50Hz. However, the IMU generates outputs at four times this rate, namely 200Hz. To accommodate this bearing in mind that the controller is implemented at 50Hz, the incremental positions and velocities are divided by four and the FPGA transmits the same information four times during the main 50Hz clock cycle. The 50Hz clock rate is chosen since that is the expected closed-loop sampling rate on the actual helicopter. All components of the HITL simulator have been tested and the overall system has been integrated (Figure 3).

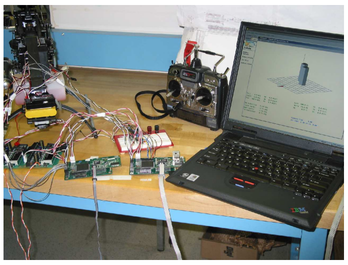


Fig. 3. Hardware-in-the-loop simulator.

5. CONTROL DESIGN AND IMPLEMENTATION ON AUTO-PILOT

Figure 4 shows the structure of the controller to track a given position reference trajectory (x_d, y_d, z_d) with a prescribed yaw z_d (usually π). The reference trajectory is generated by the GODZILA path-planning and obstacle-avoidance algorithm described in Section 6. The control scheme in Figure 4 can be interpreted as a backstepping-based controller which synthesizes pseudo-commands for the roll and pitch which serve as the *virtual control inputs* for the planar (x, y) dynamics. The pitch cyclic and roll cyclic control inputs are then designed to regulate the pitch and roll to their designed references. The simplest version of this controller is with each of the blocks being a PD/PID controller. Gain-scheduling and/or nonlinearities can be incorporated to add robustness. Furthermore, saturations are used to ensure that the pitch and roll pseudo-commands are physically realizable and that the control commands are within the servo limits. The performance of the controller for hover is illustrated in Figures 5 and 6.

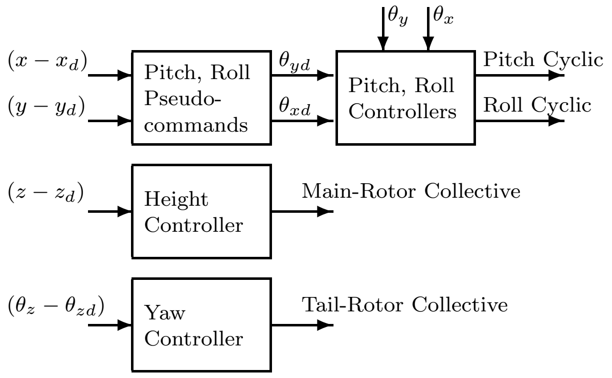


Fig. 4. Controller structure.

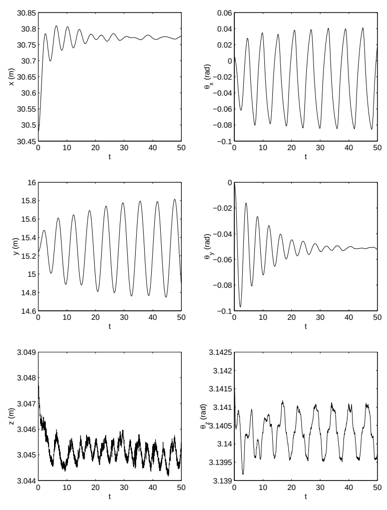


Fig. 5. Hover: positions and rotations.

6. GODZILA PATH-PLANNING AND OBSTACLE-AVOIDANCE ALGORITHM

GODZILA (Game-Theoretic Optimal Deformable Zone with Inertia and Local Approach) is a path-planning and obstacle avoidance algorithm (Krishnamurthy and Khorrami) for navigation in completely unknown environments without requiring the building of an obstacle map. The algorithm follows a purely local approach using only the current range sensor measurements at each sampling instant and requiring only a small number of stored variables in memory. This minimizes

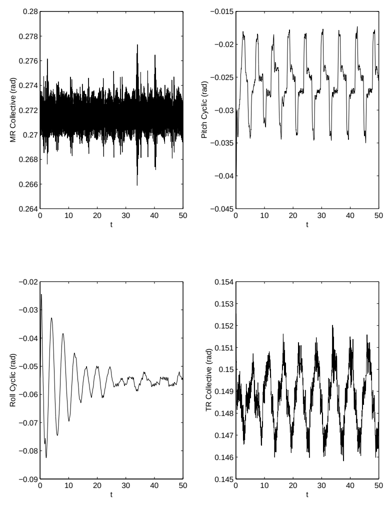


Fig. 6. Hover: control inputs.

the memory and computational requirements for implementation of the algorithm, a feature that is especially attractive for small autonomous vehicles.

The trajectory is generated through online solution of an optimization cost at each sampling instant. The optimization cost can be chosen so that the minimizer is obtained in closed form. The optimization cost has three terms penalizing, respectively, motion in directions other than the direction to the target, motion towards obstacles, and back-tracking. By appropriately weighting the terms, a specified clearance to obstacles can be ensured. In addition to the optimization algorithm, GODZILA includes two components, a local straight-line planner utilized if the target is visible and navigation towards a random target. Since the algorithm follows a local approach, it is possible to be caught in a local limit cycle oscillation or *trap* which can be detected using, for instance, the variance of the position variable over some number of successive sampling instants. When a trap is detected, navigation towards a randomly chosen target is initiated to escape the trap. It is proved in (Krishnamurthy and Khorrami) that GODZILA provides convergence to the target in finite time with probability 1 in any finite-dimensional space.

The performance of GODZILA is demonstrated through simulations in Figures 7-10. The number of range sensors used in the two-dimensional simulations is three with the orientations being at angles of 0^o, 45^o and -45^o with respect to the vehicle heading. The three-dimensional simulations use five range sensors oriented at yaw and pitch of $(0^o, 0^o)$, $(-45^o, 0^o)$, $(45^o, 0^o)$, $(0^o, 45^o)$, and $(0^o, -45^o)$ with respect to the current heading. Simulations with two simple 2D environments are shown in Figure 7. In Figure 8, a more complicated obstacle map is shown in which the path to the target has to pass through a narrow corridor which is visible only from a small region in the space. Note that GODZILA uses only the current sensor measurements and avoids building a map of the environment. Hence, a period of wandering is seen since no way out of the enclosed space is initially visible. However, when a local trap is detected, random navigations are initiated which successfully bring the vehicle into the region from which the opening is visible.

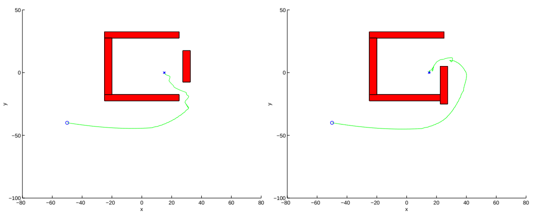


Fig. 7. Two simple 2D obstacle maps.

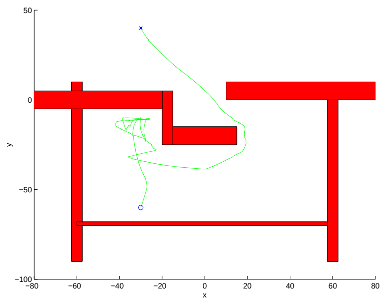
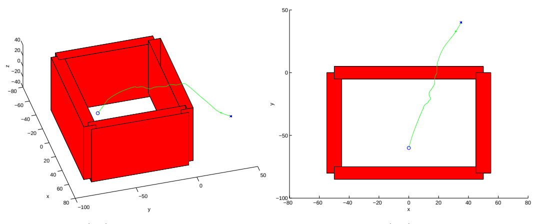


Fig. 8. A more complicated 2D maze.

Figure 9 shows a 3D simulation in which the vehicle starts from an enclosed region to escape from which it needs to move either above or below the wall. In Figure 10, a small window is provided and it is seen that the algorithm elects to fly through the window leading to a shorter path.



(a) (b)

Fig. 9. 3D simulation: Vehicle flies over wall to reach target. (a) 3D view; (b) Top view.

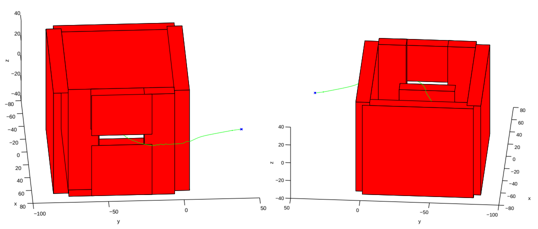


Fig. 10. 3D simulation: Vehicle flies through a window; two 3D views.

REFERENCES

Bendotti, P. and J.C. Morris (1995). Robust hover control for a model helicopter. In: *American Control Conference*. Seattle, WA. pp. 682–687.

Civita, M. La, G. Papageorgiou, W. C. Messner and T. Kanade (2002). Design and flight testing of a gain-scheduled H_∞ loop shaping controller for wide-envelope flight of a robotic helicopter. In: *American Control Conference*. Denver, CO. pp. 4195–4200.

Frazzoli, E., M.A. Dahleh and E. Feron (2000). Trajectory tracking control design for autonomous helicopters using backstepping algorithm. In: *American Control Conference*. Chicago, IL. pp. 4102–4107.

Gavrilets, V., I. Martinos, B. Mettler and E. Feron (2002). Control logic for automated aerobatic flight of a miniature helicopter. In: *AIAA Guidance, Navigation and Control Conference*. Monterey, CA.

Heffley, R.K. and M.A. Mnich (1988). Minimum complexity helicopter simulation math model. Technical Report NASA Contract Report 177476. NASA.

Isidori, A., L. Marconi and A. Serrani (2003). Robust nonlinear motion control of a helicopter. *IEEE Trans. on Automatic Control* **48**(3), 413–426.

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

Koo, T.J. and S. Sastry (1998). Output tracking control design of a helicopter model based on approximate linearization. In: *IEEE Conference on Decision and Control*. Tampa, FL. pp. 3635–3640.

Krishnamurthy, P. and F. Khorrami (2005). GODZILA: A low-resource algorithm for path planning in unknown environments. In: *American Control Conference*. Portland, OR.

Kung, C.C., C.D. Yang, D.W. Chiou and C.C. Luo (2002). Nonlinear H_∞ helicopter control. In: *IEEE Conference on Decision and Control*. Las Vegas, NV. pp. 4468–4473.

Mazenc, F., R. E. Mahony and R. Lozano (2003). Forwarding control of a scale model autonomous helicopter: A lyapunov control design. In: *IEEE Conference on Decision and Control*. Maui, HI. pp. 3960–3965.

Shim, H., T.J. Koo, F. Hoffmann and S. Sastry (1998). A comprehensive study of control design for an autonomous helicopter. In: *IEEE Conference on Decision and Control*. Tampa, FL. pp. 3653–3658.

Yue, A. and I. Postlethwaite (1990). Improvement of helicopter handling qualities using H_∞-optimisation. *IEE Proceedings* **137**(3), 115–129.