

FLIGHT SIMULATION FOR THE DEVELOPMENT OF AN EXPERIMENTAL UAV

Eric N. Johnson* and Sumit Mishra†

School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150

ABSTRACT

The use of flight simulation tools to reduce the schedule, risk, and required amount of flight testing for complex aerospace systems is a well-recognized benefit of these approaches. However, some special challenges arise when one attempts to obtain these benefits for the development and operation of an Experimental Uninhabited Aerial Vehicle (UAV) system. These types of UAV systems are characterized by the need for the need for continual checkout of experimental software and hardware. Also, flight-testing can be further leveraged by complementing research results with flight-test validated simulation results for the same experimental UAV. In this paper, flight simulation architectures for system design, integration, and operation of an experimental helicopter-based UAV, the GTMax, are explored, and the development of a simulation tool described. The chosen helicopter-based UAV platform (a Yamaha R-Max) is well instrumented: differential GPS, inertial measurement unit, sonar altimeter, radar altimeter, and a 3-axis magnetometer. One or two flight processors can be utilized.

INTRODUCTION

This paper describes the development and use of simulation tools for the Georgia Tech R-Max VTOL UAV (GTMax), shown in Figure 1, and related UAV programs. The GTMax is a highly capable platform for UAV research that has been developed over the past 12 months, and builds on previous UAV work at Georgia Tech. It has already been used to accomplish important research objectives and to compete in an international engineering competition. The GTMax system will first be described in limited detail. Then, the simulation tools developed to support its development and use will be described, including a variety of configurations that have been used. Following this, results and conclusions are discussed.

* Lockheed Martin Assistant Professor of Avionics Integration, Member AIAA.

E-mail: Eric.Johnson@aerospace.gatech.edu

† Graduate Research Assistant

E-mail: Sumit_Mishra@isye.gatech.edu



Figure 1 – GTMax UAV

GTMax UAV

The GTMax is based on the Yamaha R-Max industrial helicopter airframe, which has the following characteristics:

- Rotor Diameter: 10.2 ft, Length: 11.9 ft
- Gross Weight: 205 lb, Payload: >66 lb
- Gasoline, 2 Cylinder, Water Cooled, 246cc displacement, 21 hp
- Endurance of 60 min
- Generator and Battery, 12V
- Electric Starter

The hardware components that make up the basic flight avionics include general purpose processing capabilities and sensing, and add about 35 lbs to the basic airframe. The interface to the helicopter is via a modified Yamaha Attitude Control System (YACS) interface that allows raw servo commands to be given. The current configuration includes:

- 266MHz Embedded PC, 500 Mb Flash Drive
- 850 MHz Embedded PCs, 500 Mb Flash Drive
- ISIS Inertial Measurement Unit
- NovAtel RT-2 Differential GPS
- 3-Axis Magnetometer
- Sonar Altimeter
- Radar Altimeter

- Vehicle Telemetry (RPM, Voltage, Pilot Inputs) from YACS
- Actuator control to YACS
- 11 Mbps Ethernet Data Link
- RS-232 Serial Data Link
- Axis Video Server
- CCD Camera

These components have been packaged into exchangeable modules: 2 computer modules, GPS module, data link module, and IMU/Radar module. These modules are placed in a vibration-isolated rack below the main body of the helicopter, shown in Figure 2. There is also a sonar/magnetometer assembly at the tail, and a power distribution system including circuit breakers near the module rack. Each module has its own self-contained power regulation and Electro-Magnetic Interference (EMI) shielding.



Figure 2- Vibration Isolated Avionics Module Rack

The power distribution system utilizes the onboard generator, which outputs 12V DC, 10 A. It includes a hot-swappable connection to use external power. Each component has a dedicated individual circuit breaker.

Wiring external to the modules consists of RS-232 Serial, Ethernet, and 12V DC only. Wiring is routed to one side of the module rack, Figure 3. The other side is kept free, Figure 2, and available for temporary hookups (e.g. Ethernet) and status LEDs. The complete wiring diagram is shown in Figure 4, including RS-232, Ethernet, and power wiring. Note the compartmentalization in modules, dedicated power regulation within each module, and the interface to the YACS.



Figure 3 – Wiring on One Side of Module Rack

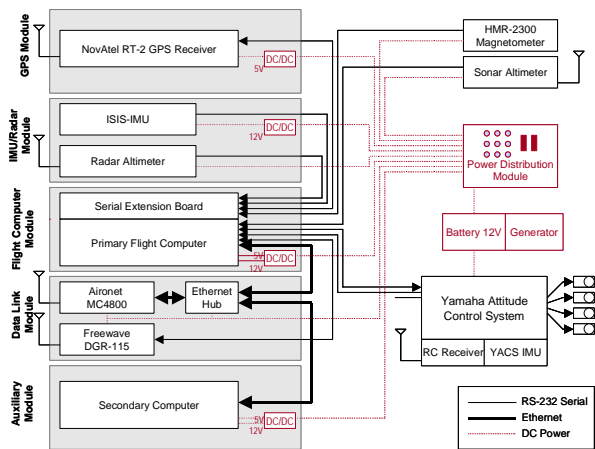


Figure 4: GTMax Wiring Diagram

The operating systems utilized for the onboard (Flight) software are either VxWorks, QNX, Linux, or a combination. Operating system independence is maintained to maximize the ability to support varied research programs. The operating system independence is accomplished by extensive use of ANSI C/C++ and the OpenGL API. No special compilers are utilized. Normally Microsoft Visual Studio is used for Windows and gcc is used for Linux and QNX.

The onboard software runs on the primary flight computer or secondary computer. The Ground Control Station (GCS) software runs on the ground, normally on one or more laptop computers. The simulation software is not used in the flight configuration. All of the above software is included in the GCS or simulation builds. Only the onboard software is included in a primary flight computer or secondary computer build.

The baseline navigation system running on the primary flight computer is a 17 state Extended Kalman Filter. The states include: vehicle position, velocity, attitude (quaternion), accelerometer biases, gyro biases,

terrain height. The system is all attitude capable and updates all states at 100 Hz¹.

The baseline controller is an adaptive neural network trajectory following controller with 16 Inputs, 5 hidden layer neurons, and 6 outputs for each of the 6 degrees of freedom². The controller can also be configured as a conventional inverting controller.

Data Communication

Generic and highly capable data communication software has been developed to support a large number of potential flight and simulator configurations. First and foremost, these routines supports serial data reading and writing as necessary for the Commercial Off The Shelf (COTS) sensors and other components used.

These same routines can also be used to re-route this data through Ethernet or as memory within an single executable. This data can be re-routed in real-time, in software. It should be noted that almost all operating system specific software is limited to these routines.

These routines are used to: interface with all sensors, the wireless serial datalink, and to repeat all wireless serial data over the wireless Ethernet (for redundancy). Additionally, any data received over a link can be stored to a binary file. This recorded data can then be played back to stimulate selected components.

Simulation Tools

Early in the program, the top-level simulation requirements to support the development and operation of an experimental UAV described above were identified as:

1. Test all custom developed software and guidance, navigation, and control algorithms (more extensively than practical or safe in a flight test setting)
2. Test onboard computer hardware, operating system implementation, and software execution in real-time
3. Rehearse all procedures and flight test plans
4. Visualize recorded flight test data
5. Reconstruction of flight test events after-the-fact (e.g., incident reconstruction)
6. Can be utilized at the flight test location
7. Supports flight test data validation

From these, the following additional requirements were derived:

1. Models of the sensors, aircraft, and aircraft interfaces – down to the level of binary serial packets
2. Injection of model error and environmental disturbances
3. Scene generation capability
4. Can operate effectively using a single laptop computer
5. Reconfigurable data communication routines discussed above

The simulator tool that has been developed normally runs on a high-end personal computer or laptop that uses the Windows 2000/NT operating system or Linux, and is written primarily in C. It includes an aircraft model, the aircraft interface model, and sensor models. The aircraft model has six rigid-body degrees of freedom plus engine, fuel, and rotor dynamics. The helicopter interface model simulates the servo interface unit functionality and RS-232 serial interface. The sensor models include errors, mounting location and orientation, latency, and digital interfaces. The scene generator is a real-time 3D graphics window, Figure 5, showing the aircraft and the terrain, and has additional functionality to aid in data visualization.



Figure 5 – Simulator Scene Generator

The basic simulation tools allows for real-time display of all data, including plotting and logging. It also allows one to modify any data. The simulator can run in real time or in a batch mode (faster than real time).

To utilize the Software-In-The-Loop (SITL) configuration, Figure 6, the un-compiled software source code, which normally runs on the onboard computer, is compiled into the simulation tool itself, allowing this software to be tested on the simulation host computer. This allows the flight software to be

tested without the need to tie-up the flight hardware. Once any modification has been tested with the SITL configuration, the Hardware-In-The-Loop (HITL) simulation configuration is used, Figure 7. For the HITL simulation, the onboard computer is plugged into a simulation-host computer. Here, the hardware under test is the onboard computer(s), servos, along with all software that executes on the computer(s). The sensor and helicopter interface models provide the proper interfaces to the onboard computer, so the onboard computer configuration is identical to that used in a flight test. The HITL simulation is used to test all guidance, navigation, and control algorithms software and as much of the hardware as practical, in real-time. This configuration is used each time any modification is made concerning either the onboard software or the onboard computer hardware, and prior to any new flight test activity.

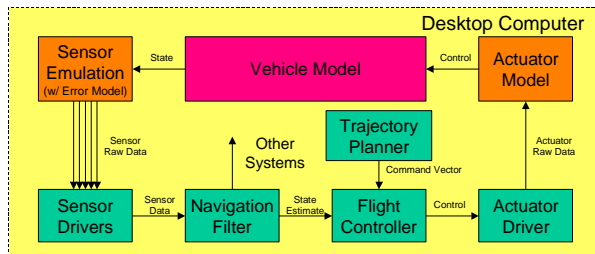


Figure 6 – Software in the Loop Structure

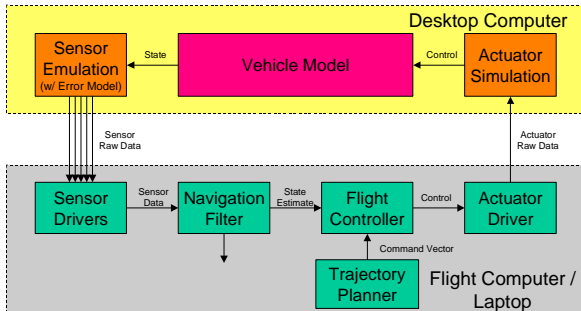


Figure 7 – Hardware in the Loop Structure

An alternate for of the SITL configuration is shown in Figure 8, where the Ground Control Station (GCS) interface is also included and run. This allows an operator interact with the vehicle in the same way it is used in flight. The GCS and onboard software/vehicle models can be run from a single executable, as separate processes on a single CPU, or on two separate machines on a network. This allows all the details of communicating data between the GCS and the onboard computers to be tested thoroughly before tying up flight hardware.

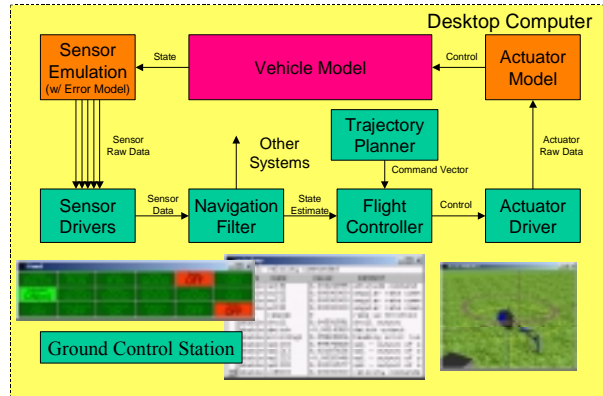


Figure 8 – Alternate SITL configuration which includes GCS interface

Other Simulator Configurations Used

Truck testing of navigation system: This configuration was used to test the onboard navigation software operating with the actual sensors without flying the vehicle. This was accomplished first statically, and then dynamically in a pickup truck. For some of these tests a laptop acted as the onboard computer (enabling greater access to data, since the GCS and onboard software were running on the same machine) and later the actual flight computer was used. The final test was done with the helicopter itself on the truck, shown in Figure 9.



Figure 9 – Truck testing of Navigation System

Fake GPS data in the lab: Prior to a flight test day, it is important to bring up the flight software in the flight configuration to test basic functionality statically. Because this is usually done indoors, there is no GPS data available. To facilitate this test, simulated GPS data is sent from a simulator running on some other machine on the network to replace data coming from the actual GPS receiver. This configuration is illustrated in Figure 10.

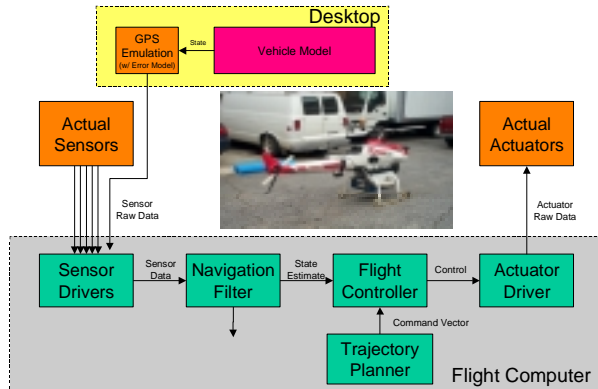


Figure 10 – Fake GPS Data for Testing Indoors

Image processing configurations: A recent mission performed by the GTMax for the International Aerial Robotics Competition required several simulator configurations to support elements of the required image processing subsystem. The first configuration used allowed still images (from flight or other source) to be send one-at-a-time to the image processing subsystem running as part of the SITL configuration (on one or more networked machines), Figure 11.



Figure 11 – Onboard Single Image After Processing, Red box Indicates a Correctly Located Building

The second configuration utilized recorded video data from a flight test. To accomplish this, video was played back and sent to the onboard video server (the flight hardware) and then to the image processor. The image processor was either the flight hardware itself or an alternate networked machine, or both.

The third and final image processing configuration was used to test the tracking and mapping logic used. Here, the simulator generated fake video for onboard

video server. One way this was configured was to place a laptop computer in front of the GTMax onboard camera, utilizing the flight hardware for the camera, video server, and image processing computer. This is shown in Figure 12.



Figure 12 – One of the Image Processing Test Configurations, Simulated Images are Being Shown to the GTMax Onboard Camera

Navigation data playback: Once onboard raw sensor data had been recorded in flight, extensive use of this data was made in improving the navigation system. This was done by playing back this data in real time or faster than real time, and executing the onboard navigation flight software.

GCS data playback: As stated above, the communication routines can be configured to save and playback data sent over a serial line. All data received from the helicopter in flight is recorded in this manner, allowing for reconstruction of events in the case where all other data was lost.

RESULTS AND CONCLUSIONS

The extensive use of a variety of simulation configurations has been of considerable benefit for the recent development of the GTMax UAV, and for its use in research and the International Aerial Robotics Competition. The key features of a flexible data communication system, models for all hardware components, and a simulation software infrastructure enable these configurations. The benefits have included increased safety, effective participation of a large number of researchers, the detection of errors before flight testing, and the effective use of flight test data.

ACKNOWLEDGEMENTS

The authors would like to acknowledge some of the many other contributors to this work: Henrik Christophersen, J. Eric Corban, Cameron Craddock,

Manuj Dhingra, Joerg Dittrich, Richard Guily, Jincheol Ha, Bonnie Heck, Jeong Hur, Suresh Kannan, Wayne Pickell, JVR Prasad, Alison Proctor, Daniel Schrage, George Vachtsevanos, and Linda Wills.

REFERENCES

¹Dittrich, J., and Johnson, E., “Multi-Sensor Navigation System for an Autonomous Helicopter,” *Digital Avionics Systems Conference*, 2002.

²Johnson, E. and Kannan, S., “Adaptive Flight Control for an Autonomous Unmanned Helicopter,” *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.