

# UAV Flight Test Programs at Georgia Tech

Eric N. Johnson<sup>\*</sup>, Daniel P. Schrage<sup>†</sup>, J.V.R. Prasad<sup>‡</sup>, and George J. Vachtsevanos<sup>‡</sup>  
*Georgia Institute of Technology, Atlanta, GA 30332-0150*

**This paper describes the design, development, and operation of research Unmanned Aerial Vehicles (UAVs) that have been developed at the Georgia Institute of Technology in the Schools of Aerospace and Electrical/Computer Engineering. This includes a description of flight test experiences and lessons learned over the past 10 years, with emphasis on recent work. Specifically in 1998, Georgia Tech acquired a Yamaha R-Max Remotely Piloted Helicopter (RPH) for use in the DARPA Software Enabled Control and other research programs. An open system UAV testbed was developed based on this vehicle that is referred to as the GTMax. The use of a variety of simulation configurations has been of considerable benefit. Flexible data communication systems, models for all hardware components, and a flexible simulation software infrastructure are important. The use of a modular avionics architecture and the use of a vehicle with a relatively large payload capacity also allow these UAVs to be configured quickly for a variety of experiments.**

## 1. Introduction

This paper describes the design, development, and operation of research Unmanned Aerial Vehicles (UAVs) that have been developed and operated at the Georgia Institute of Technology. This includes a description of flight test experiences and lessons learned over the past 10 years, with emphasis on recent work performed in the Schools of Aerospace and Electrical/Computer Engineering. The objectives of much of this work relates to increasing the reliability and performance of future unmanned systems, including technologies for increased autonomy. This experience has included involvement in the Association for Unmanned Vehicle Systems, International (AUVSI) International Aerial Robotics Competition, including winning it in 1993 with the first demonstration of autonomous flight of an unmanned helicopter, including takeoff and landing, at this contest<sup>1</sup>. This was followed by the U.S. Army Autonomous Scout Rotorcraft Testbed (ASRT) project from 1994 to 1997<sup>2,3</sup>, Figure 1.



*Figure 1 - ASRT vehicle weighed approximately 35 pounds*

In 1997 Georgia Tech purchased two Yamaha R-50 remotely piloted helicopters (RPHs) for use in flight controls research under the Army/NASA sponsored Georgia Tech Center of Excellence in Rotorcraft Technology (CERT) program, Figure 2. Flight control technologies, such as neural network adaptive flight control, tested on these RPHs have been transferred to Boeing for flight tests on the X-36 and JDAM programs as well as to NASA Ames and MSFC for use in simulation studies<sup>4,5</sup>. Other researchers have also had success utilizing Yamaha helicopters<sup>6-8</sup>.

---

<sup>\*</sup> Lockheed Martin Assistant Professor of Avionics Integration, AIAA member, Eric.Johnson@ae.gatech.edu

<sup>†</sup> Professor, Fellow AIAA

<sup>‡</sup> Professor



*Figure 2 – Yamaha R-50 in Georgia Tech research configuration, operated in partnership with Guided Systems Technologies, Inc.*

In 1998 Georgia Tech began work under the DARPA Software Enabled Control (SEC) program for research and development of an Open Control Platform (OCP) and for on-line control algorithms<sup>9</sup>. Georgia Tech then acquired a Yamaha R-Max RPH, with twice the payload of the R-50s, for use in the SEC and other research and development programs. Georgia Tech has developed an open system UAV testbed based on this platform. This open system UAV testbed is referred to as the GTMax, Figure 3, and includes four major elements. These are the basic Yamaha R-Max RPH, a modular avionics system, the SEC OCP (being developed jointly with Boeing Phantom Works) with baseline guidance/navigation/control components (software), and a set of simulation tools.



*Figure 3 – GTMax Research UAV*

In January 2002 Georgia Tech was chosen to be the SEC rotary wing experiments lead. This included working with the other SEC technology developers in integrating and demonstrating their SEC technologies on the GTMax. A unique integrated simulation and flight testing approach was developed to support these activities. It includes a smooth transition from Software-In-The-Loop (SITL) to Hardware-In-The-Loop (HITL) simulation, followed by flight testing. This also included the development of a smaller research UAV, based on the MASS HeliSpy 11 inch ducted-fan vehicle, Figure 4. An SEC benchmark demonstration was completed in May 2002 and a series of planned mid-term and final experiments conducted over the next two years, culminating in a final demonstration in August 2004.



*Figure 4 – HeliSpy 11 inch ducted fan vehicle*

Georgia Tech also has done work with fixed-wing research aircraft, including a ¼ scale Piper Cub utilized in 2001 in the Aerial Robotics Competition, Figure 5, and gliders used for vision-based control, Figure 6. Also, a two larger airplane UAVs are currently under construction with similar payload and endurance capabilities to the GTMax system.



*Figure 5 – ¼ Scale Piper Cub used for 2001 Aerial Robotics Competition*



*Figure 6 – Gliders used for vision-based control*

This paper describes the development of these systems, concentrating on the GTMax first – it being the most capable of current systems. Following this, the use of simulation and software tools is described. This is following by an overview of experience applying these methods to other research vehicles. Following this, selected lessons learned and conclusions are presented.

## 2. GTMax Research UAV

### A. Hardware Elements

The GTMax utilizes the Yamaha R-Max industrial helicopter airframe, which has a 10.2 ft rotor, a maximum gross weight of 205 pounds. It has a 21 hp engine, and an endurance of approximately 60 minutes.

The hardware components that make up the basic flight avionics include general purpose processing capabilities and sensing, and add approximately 35 lbs to the basic airframe (for a total weight of 157 pounds as currently configured). The interface to the helicopter is via a modified Yamaha Attitude Control System (YACS) interface that allows raw servo commands to be given without modification by the YACS. The current configuration includes:

- 266MHz Embedded PC, 500 Mb Flash Drive
- 850 MHz Embedded PC, 500 Mb Flash Drive
- ISIS Inertial Measurement Unit (IMU) (3 accelerometers, 3 rate gyros)
- NovAtel RT-2 Differential GPS (DGPS)
- 3-Axis magnetometer
- Sonar altimeter
- Vehicle telemetry (RPM, Voltage, Remote Pilot Inputs) from YACS
- Actuator control interface to YACS
- 11 Mbps Ethernet data link and an Ethernet switch
- RS-232 serial data link
- Axis video server and CCD camera
- Axis pan/tilt/zoom camera

These components have been packaged into exchangeable modules: 2 computer modules, the GPS module, the data link module (wireless Ethernet, wireless serial, Ethernet switch), and the IMU module. These modules are placed in a vibration-isolated rack below the main body of the helicopter, shown in Figure 2. Each module has its own self-contained power regulation, air-cooling, and Electro-Magnetic Interference (EMI) shielding. There is also a sonar/magnetometer assembly at the tail, a power distribution system including circuit breakers near the module rack, and mounting points for camera systems and other components under the nose. The power distribution system utilizes the onboard generator, which outputs 12V DC. 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 7, and available for temporary hookups (e.g. Ethernet), status LEDs, and switches. The complete wiring diagram is shown in Figure 8, including a typical configuration of RS-232, Ethernet, and power wiring. Note the compartmentalization in modules, dedicated power regulation within each module, and the interface to the YACS via multiple serial lines.



Figure 7 - Vibration isolated avionics module rack, from left to right: data link module, GPS module, computer module #1 (flight computer), computer module #2 (auxiliary)

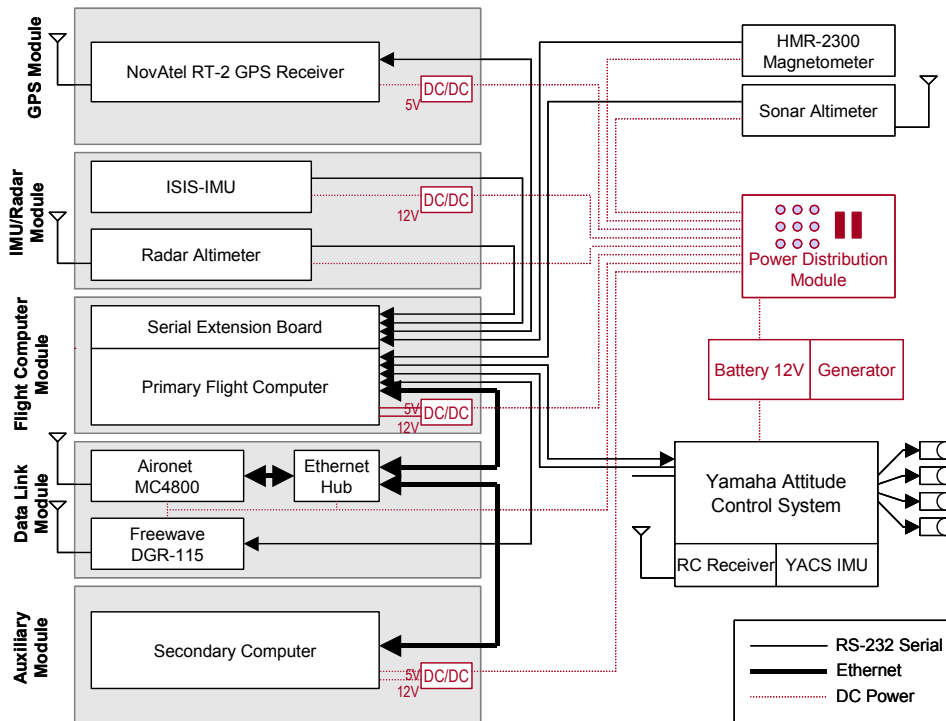


Figure 8 - GTMax wiring diagram, including separation into modules, each with their own power regulation, air-cooling, and EMI shielding

## B. Software Elements

The operating systems utilized for typical onboard (Flight) software are: 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 for graphics used in simulations and graphical user interfaces). No special compilers are utilized. Normally Microsoft Visual Studio is used for Windows and the GNU c-compiler is used for Linux and QNX. The onboard software runs on the primary flight computer and/or secondary computer. The Ground Control Station (GCS) software runs on the ground, normally on one or more laptop computers, and is used to for human operators to interact with the onboard systems. Simulation-specific software is that not used in the flight configuration. All of the above software is included in the GCS or simulation-tool 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, and terrain height error. The system is all-attitude capable and updates at 100 Hz<sup>10</sup>. The baseline flight controller is an adaptive neural network trajectory following controller with 18 neural network inputs, 5 hidden layer neurons, and 7 outputs for each of the 7 degrees of freedom<sup>11,12</sup>. The 7 degrees of freedom include the usual 6 rigid-body degrees of freedom plus a degree of freedom for rotor RPM. The controller can also be configured as a conventional inverting controller. The baseline flight controller and navigation system, which coupled with the simple baseline trajectory generator, is capable of automatic takeoff, landing, hover, flight up to the maximum attainable by the helicopter (around 85 feet/sec) and aggressive maneuvering, discussed further in the results section below.

Generic and highly capable data communication software has been developed to support a large number of potential flight and simulator test configurations. First, these routines supports serial data reading and writing as necessary for the Commercial Off The Shelf (COTS) sensors and other custom components used. These same routines can also be used to re-route any data through Ethernet or as memory within a single executable. These data

routings can be modified in real-time, by software switch. 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 data link, and to repeat all wireless serial data over the wireless Ethernet (for redundancy). Also, 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. All data received from the helicopter over either data link is stored in this manner during flight.

### C. Results

More than 200 research flight tests have been conducted on the GTMax, including tests of the baseline guidance, navigation, and control algorithms, SEC program tests, and the 2002 and 2003 Aerial Robotics Competitions. Video of many of these tests can be found at: [uav.ae.gatech.edu](http://uav.ae.gatech.edu).

The baseline guidance, navigation, and control system has been used extensively in over 200 research test flights over more than two years of operation. This has included automatic flight up to the maximum attainable level-flight speed of the machine. It has also included automatic takeoffs and landings. Depending on whether the helicopter is attempting a takeoff or a landing/hold, the rotor RPM is ramped either up to flight speed or down to idle respectively. Takeoffs are performed by ramping rotor RPM and collective until the helicopter is detected airborne, at which point the trajectory generator produces a smooth climb trajectory. Landings end with a slow vertical descent command until ground contact is detected and rotor RPM and collective pitch are reduced to an idle state. To date, approximately 100 landings and 50 takeoffs have been completed in this way.

An automatic flight envelope protection system that utilizes an online trained neural network to predict and then avoid flight envelope limits. Flight tests conducted to date include avoidance of a rotor stall prediction parameter (Erits factor, in units of speed), set artificially conservative to facilitate safe testing<sup>13</sup>.

The scenario of a stuck collective pitch actuator was simulated in flight by limiting the deflection of swash plate actuators in such a way to prevent changes in collective pitch. A fault tolerant control module was developed, running with the OCP on the 2<sup>nd</sup> flight computer, generated a rotor RPM command that allows the existing flight controller to continue to function, albeit at reduced capability. This capability could be enough to safely recover the vehicle in such a scenario. The range of acceptable rotor RPM command was experimentally determined to be 700 to 950, all utilizing the baseline adaptive flight controller without modification (normally rotor RPM is 850). Note that flight at 700 RPM required saturated collective pitch in order to hover.

For the aerial robotics competition, automated capabilities were added to the GTMax that include: searching a prescribed area, identifying a specific building within that area based on a small sign located on one wall, and then identifying an opening into that building was developed and tested. Results include successful evaluation at the McKenna Military Operations in Urban Terrain flight test site. In a separate/related activity, the image processor used to locate windows was successfully used to update an inertial navigation solution, allowing the vehicle to operate without GPS or other aiding for extended periods when trackable objects are in view.

### 3. Simulation Tools

Early in the GTMax system design, the top-level simulation requirements to support the development and operation of an experimental UAV were identified as:

1. Test all custom developed software and guidance, navigation, and control algorithms in a rigorous manner (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. Flight test data validation

All of these capabilities have subsequently been used extensively, and are considered important. From these, the following component requirements were derived:

1. Models of the sensors, aircraft, and aircraft interfaces – down to the level of binary serial data (i.e., packets) with time delays
2. Injection of model error and environmental disturbances, of those expected to be experienced in flight and worse
3. Flexible scene generation capability
4. Can operate effectively using a single conventional laptop computer
5. Reconfigurable data communication routines

The simulator tools that were subsequently developed normally run on a high-end personal computers or laptops that uses the Windows operating system or Linux, and is written primarily in C/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, time delays, and digital interfaces. The scene generator is a real-time 3-D graphics window, Figure 9, showing the aircraft and the terrain, and has additional functionality to aid in data visualization or use in the Ground Control Station<sup>14</sup>.

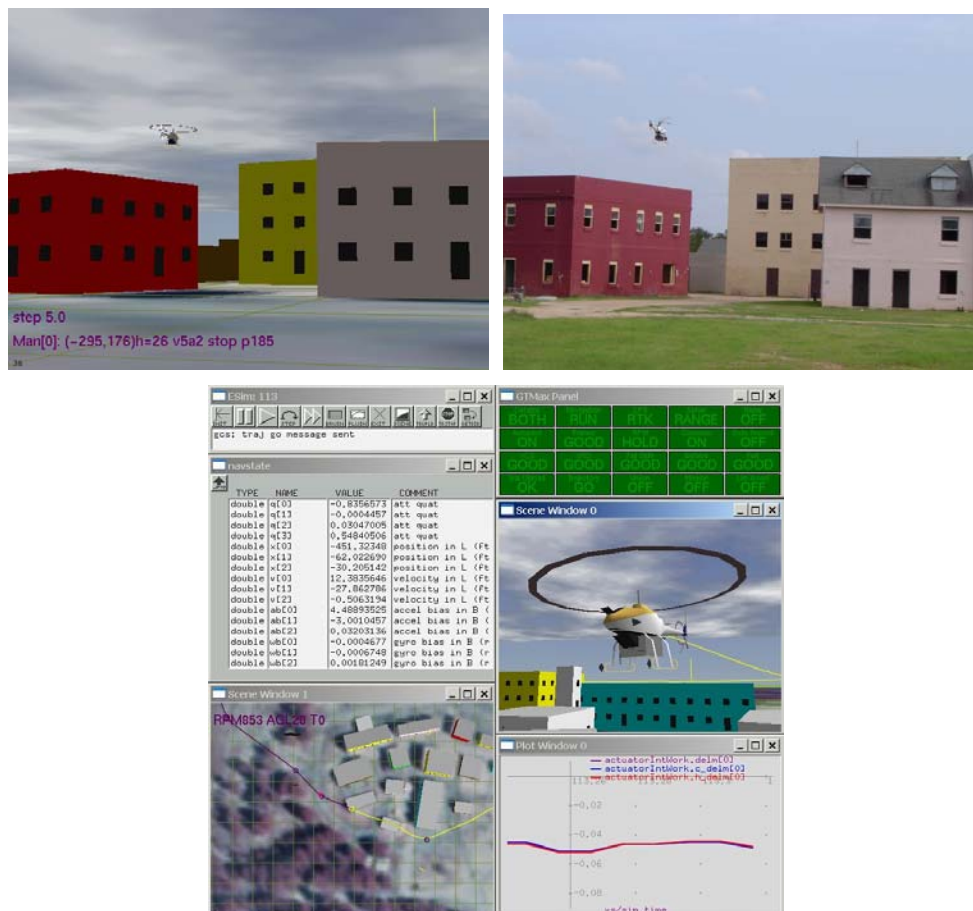


Figure 9 – Simulator scene generator (above-left) with corresponding real-world picture (above-right) and its use in the Ground Control Station (GCS) interface (below)

The scene window includes several vehicle viewing options like cockpit view, chase view, ground view, ground track, or camera view. Furthermore, visualization aids like common vehicle reference frames, trajectory, and head-up display can be selected. The scene window also allows for mission planning. The user can enter waypoints with position, velocity, and acceleration constraints. The resulting projected trajectory is then displayed in the scene. The figure also shows a status window that shows major status highlights such as automatic flight, the status of GPS lock and other salient information that may be useful during a flight test. With correct configuration of the executables on various computers using scripts, SITL and HITL configurations may be tested quickly. During a flight test, one or more laptops on the ground are configured as ground stations with data links and telemetry.

The basic simulation tools allow for real-time display of all onboard data, including plotting and logging. With time, the system has evolved to have more-and-more flexibility to support research operations. Specifically, it now allows one to view or modify any variable in memory on the primary flight computer or GCS at any time. This enables researchers to quickly diagnose problems, or to obtain desired data about system performance. The simulator can run in real time or in a batch mode (faster than real time).

To utilize the common-form of the Software-In-The-Loop (SITL) simulation configuration, Figure 10, 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 any flight hardware. Once any modification has been tested with the SITL configuration, any required Hardware-In-The-Loop (HITL) simulation configurations are used. The configurations required depend on what is being tested, but a test of a change to the primary flight computer guidance, navigation, and control system is shown in Figure 11. For this 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. This HITL simulation configuration is used to test all guidance, navigation, and control algorithms software and as much of the hardware as practical, in real-time. Figure 12 shows a configuration where all of these elements are being simulated, but a second onboard computer is tested HITL.

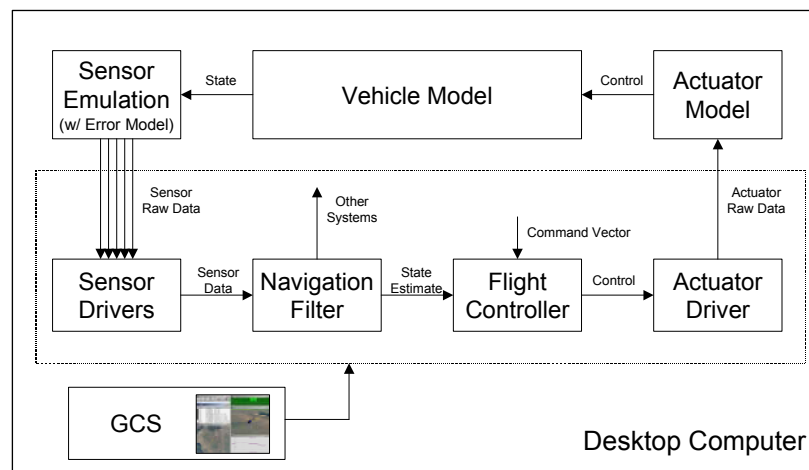


Figure 10 – Typical Software in the Loop structure, all components running on a single computer – no flight hardware is required or being tested

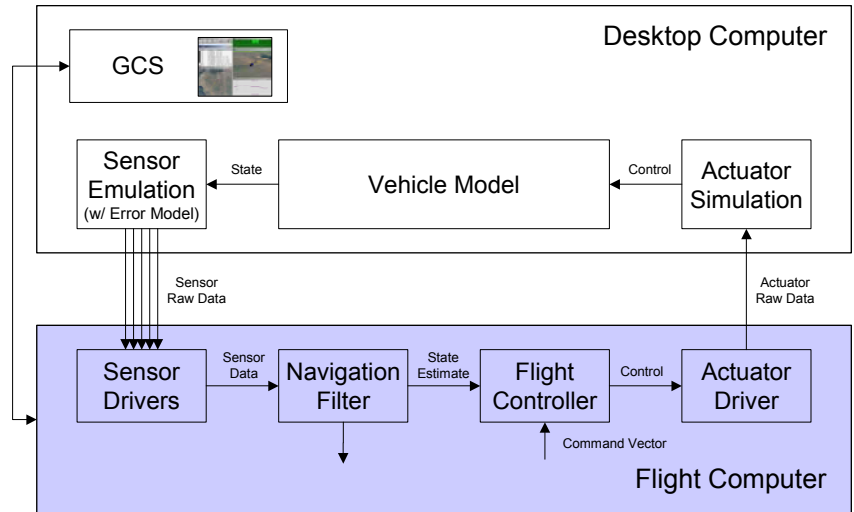


Figure 11 – Hardware in the Loop structure for testing changes to the primary flight computer software or hardware

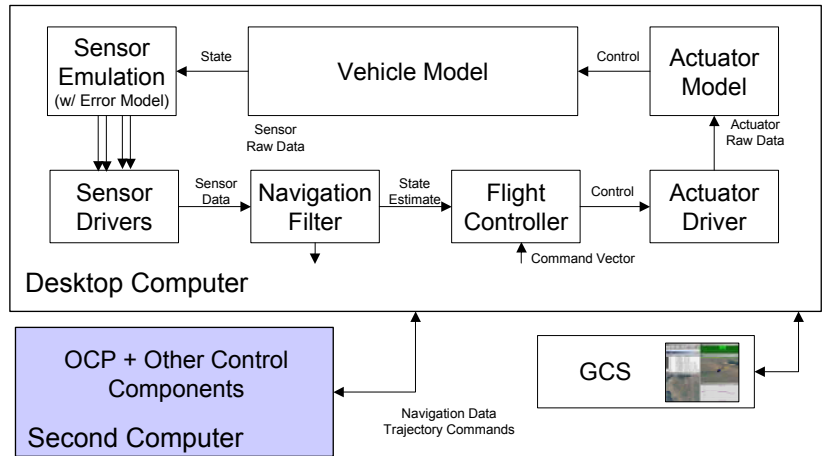


Figure 12 – Hardware in the Loop structure for testing changes to the second computer software or hardware

#### 4. Other Research UAV Programs

With the context of the GTMax and related simulation tools, this chapter describes two other flight test experiences using the previously described approaches.

##### A. Small Ducted Fan System

Recently the GTMax guidance and control algorithms were transitioned to small custom-made set of autopilot hardware, called the FCS20 illustrated in Figure 13, and tested with a MASS HeliSpy, which is an 11-inch ducted fan shown in Figure 4. The HeliSpy is a challenging vehicle because with a payload of a little over a pound. In contrast to the GTMax that has Intel-based processors and the QNX real-time operating system, the GTSpy hardware is unable to support (or chosen not to support due to various reasons such as power consumption), QNX and other hardware such as Ethernet.



Figure 13 – FCS20 miniature autopilot performs navigation, guidance, and adaptive control for small UAVs.

Operating system specific software was developed for the  $\mu$ C operating system used on the small autopilot, with some loss of functionality (e.g., no Ethernet), making a common API for this and other operating systems used (QNX, Windows, Linux) and different hardware architectures (Intel, Motorola). This allowed the entire set of baseline guidance, navigation, and adaptive control software to be utilized on the miniature autopilot – including communication capabilities.

Alternate vehicle and sensor models were added to the simulation environment to support testing this software. Following this, a number of tethered and approximately 10 un-tethered flights using this autopilot have been conducted in the summer of 2004. These flights included operations in some wind gusts (up to about 15 knots airspeed), automatic takeoffs and landings.

## B. Vision Controlled Glider

The Vision Controlled Gliders shown in Figure 6 have been used to test algorithms for flying an airplane with a conventional camera as its only sensor. In this case, it is using a single camera onboard. These images are transmitted to a frame grabber and image-processing computer on the ground. The images are processed to observe an optical target in view. The location, size, and orientation of this target is used to update a navigation solution for the glider. A simple guidance and control strategy steers the glider to the optical target, transmitting servo/actuator commands back to the glider. A Ground Control Station allows an operator to interact with the system<sup>15</sup>.

## 5. Operations

Flight test operations typically take place at either a dedicated 80 acre test location in a rural area in Georgia or at the McKenna Military Operations in Urban Terrain (MOUT) at Ft. Benning, Georgia. A customized truck is used both to transport the aircraft and as a location for the Ground Control Station equipment and operators, Figure 14. Air conditioning and power generation have been added to this truck in the past year, which has led to increased productivity during flight test operations.

Software and hardware drawings are kept track of using Concurrent Version System (CVS) configuration management software. This is critical to allow multiple developers to work on the same set of software, and to support a flight test operation with its requirements for corresponding tests with software versions.

A standard set of tests to be done prior to any flight test day are documented in a checklist to be performed prior to leaving the lab setting for the flight test field. A checklist ensures that nothing is forgotten when packing to go to the field. Finally, a checklist to be used during the flight test operation itself is tested using the simulator prior to use at the field.

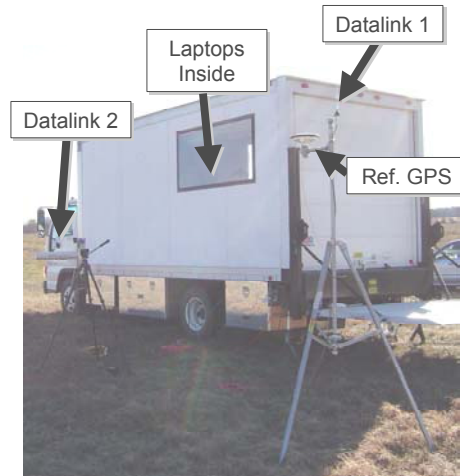


Figure 14 – Ground Control Station is usually housed in the same truck used to transport the aircraft.

## 6. Lessons Learned and Conclusions

A series of lessons learned are summarized here. They are not considered original, and in many cases reflect common aerospace industry practice. However, the nature of flight testing of innovative technologies in the context of an academic program means that we must be constantly reminded of many of these ideas to avoid re-learning them.

### A. Testing

Mitigating risk does not conflict with getting results. In fact it is the opposite. Any extra effort to ensure that a result will be safely acquired will make the result that much more likely to be obtained – and will enhance the insight gained from it. When one gets something “to work” you are at best half way done. It is important to view testing prior to flight test as a significant part of the effort required to be ready to test in flight.

Simulation tools are absolutely critical to accomplish the above items. In the context of a research flight test program, the primary purpose of flight testing should be to verify a small subset of what was experienced in simulation. Innovative testing configurations can and should be used to check critical items prior to use in flight.

The use of ground vehicles, tethers, and surrogate aircraft should be considered to create building-blocks to the ultimate flight test objective. As an example, experience with systems on the GTMax greatly sped up the development of the small autopilot used on the HeliSpy ducted fan – saving a considerable amount of effort on a much trickier platform to work with.

### B. Flexibility

Maximize the flexibility of simulation tools, this supports objectives described above. To do this, avoid software packages, as nothing is more flexible and portable than something like raw C/C++ source code. Occasionally specific tests will lead one to use certain specific software packages. You are in a much better position to be able to do that if working with your own software vs. someone else’s software package. Maintain operating system independence as much as possible, for the same reasons.

Buy good hardware. Buy hardware that can be used in many ways, and is more accurate than required. It is much easier to make a sensor “look” worse than it really is to support testing of a specific algorithm. Highly accurate sensors can be used to verify the performance of more conventional sensors.

Maximize flexibility of software, maximize the ability to monitor and modify in flight or in simulation. Avoid changing hardware or software in the field by testing and by maximizing ability to reconfigure software simply by changing parameters in memory.

### C. Standards

Software and hardware configuration management is clearly necessary. Coding standards are a second necessary condition to allow more than one person to develop effective flight software. It is important to standardize the verification and validation process when multiple versions are developed, such as in a research flight test operation.

Hardware standards from hobby airplane flying are too low for an effective research flight testing, and should be avoided when possible. Instead, those available for manned aircraft are far more appropriate when available.

Have clear roles and responsibilities for the team during flight test activities. The creativity and free-form dialog that is so important in an effective research institution must be suspended during a flight test operation and a more formal dialog should be used. Checklists should be used to ensure required items are completed. Consider briefings and de-briefings for certain tests. Always brief new people and observers about any safety items they need to be aware of.

Data recording needs to be done onboard and off board when possible. Some data should be recorded at all times in case any part-time data recorded is not engaged at a critical moment. This data should be recorded in non-volatile memory. It is important to video tape all flight operations as a backup to these recordings. The video tape can also provide important information when looking at recorded data. It is important to record relevant audio if applicable. Keeping a log of flight tests allows problems to be tracked and for one to go back and make use of previously recorded information.

A comfortable test team is a more effective one. This includes keeping hydrated, maintaining body temperature, staying dry, and having had time to sleep.

### Acknowledgements

The authors would like to acknowledge some of the many other contributors to this work: Henrik Christophersen, Scott Clements, J. Eric Corban, Graham Drozeski, Joerg Dittrich, Luis Gutierrez, Murat Guler, Bonnie Heck, Jeong Hur, Phillip Jones, Aaron Kahn, Suresh Kannan, Adrian Koller, Eric Martens, Sumit Mishra, Alex Moodie, Wayne Pickell, Alison Proctor, Suraj Unnikrishnan, Linda Wills, and Ilkay Yavrucuk. This work is supported in part by the DARPA Software Enabled Control (SEC) Program under contracts #33615-98-C-1341 and #33615-99-C-1500.

### References

- <sup>1</sup>Christophersen, H., Hart, M., Dhingra, M., Johnson, E., and Guily, R., "Development of an Autonomous Aerial Reconnaissance System at Georgia Tech," *Proceedings of the Association for Unmanned Vehicle Systems International Unmanned Systems Symposium & Exhibition*, 2002.
- <sup>2</sup>G. Vachtsevanos, W. Kim, S. Al-Hasan, M. Simon, D. Schrage and J.V.R. Prasad, "Mission Planning and Flight Control: Meeting the Challenge with Intelligent Techniques," *Journal of Advanced Computational Intelligence*, Vol. 1, No. 1, pp. 62-70, October 1997.
- <sup>3</sup>S. Al-Hasan and G. Vachtsevanos, "Intelligent Route Planning for Fast Autonomous Vehicles Operating in a Large Natural Terrain," *Journal of Robotics and Autonomous Systems*, Vol. 40, pp. 1-24, 2002.
- <sup>4</sup>Brinker, J., and Wise, K. "Flight Testing of a Reconfigurable Flight Control Law on the X-36 Tailless Fighter Aircraft." *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000.
- <sup>5</sup>Johnson, E., Calise, A., El-Shirbiny, H., and Rysdyk, R. "Feedback Linearization with Neural Network Augmentation Applied to X-33 Attitude Control." *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000.
- <sup>6</sup>O. Shakernia, C. S. Sharp, R. Vidal, D. Shim, Y. Ma, and S. S. Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle," *IEEE Int. Conference Robotics and Automation*, 2002.
- <sup>7</sup>M. La Civita, G. Papageorgiou, W. C. Messner, T. Kanade, "Design and Flight Testing of a High-Bandwidth H-infinity Loop Shaping Controller for a Robotic Helicopter," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

- <sup>8</sup>Nakanishi, H., Inoue, K., and Sato, A. "Development of Command and Control Systems for UAVs," *Proceedings of American Helicopter Society 57th Annual Forum*, 2001.
- <sup>9</sup>Wills, L., Kannan, S., Sander, S., Guler, M., Heck, B., Prasad, J.V.R., Schrage, D., and Vachtsevanos, G., "An Open Platform for Reconfigurable Control," *IEEE Control Systems Magazine*, pp. 49-64, June 2001.
- <sup>10</sup>Dittrich, J., and Johnson, E., "Multi-Sensor Navigation System for an Autonomous Helicopter," *Digital Avionics Systems Conference*, 2002.
- <sup>11</sup>Johnson, E. and Kannan, S., "Adaptive Flight Control for an Autonomous Unmanned Helicopter," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.
- <sup>12</sup>Kannan, S. and Johnson, E., "Adaptive Trajectory-based Flight Control for Autonomous Helicopters," *Proceedings of the 21st Digital Avionics Systems Conference*, 2002.
- <sup>13</sup>Yavrucuk, I., Prasad, J.V.R., Calise, A., and Unnikrishnan, S., "Adaptive Limit Control Margin Prediction and Avoidance," *58th AHS Annual Forum*, 2002.
- <sup>14</sup>Johnson, E., and Mishra, S., "Flight Simulation for the Development of an Experimental UAV," *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, 2002.
- <sup>15</sup>Proctor, A. and Johnson, E., "Vision-Only Aircraft Flight Control Methods and Test Results," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2004.