Human-in-the-Loop Distributed Simulation and Validation of Strategic Autonomous Algorithms

Christopher W. Lum,* Matthew L. Rowland,† and Rolf T. Rysdyk ‡
Autonomous Flight Systems Laboratory
University of Washington, Seattle, WA, 98105, USA

The goal of most current Unmanned Air Vehicle (UAV) research is to develop algorithms which allow a single agent or possibly a team of agents to operate completely autonomously without human interaction. Most of these autonomous algorithms operate at a high, strategic level and assume that low level tasks such as state stabilization and signal tracking have already been realized. The difficulty in verifying and validating strategic algorithms in an actual flight test is that implementing these algorithms require the development of many other lower-level subsystems which are not directly related to the strategic algorithms. This paper presents both an architecture and hardware implementation of a ground based, distributed testing environment that is used in the Autonomous Flight System Laboratory to test strategic level algorithms in an efficient manner. This system allows human interaction at very specific points to avoid developing a fully autonomous system, but still preserves the function and contributions of the strategic algorithm. This architecture and ground based testing facility greatly reduces development time and allows algorithms to be tested with little approximations before implementing them on a fully autonomous system.

Nomenclature

$\alpha^\star$	Scalar parameter used when computing pilot performance
A	Starting waypoint
B	Ending or active waypoint
d	Number of waypoints in the path
ΔT	Time step used to record data
p_i	Waypoint i
r_{max}	Distance agent can travel in a single step
t_i	Desired arrival time at waypoint p_i
$x(t)$	Agent's location at time t
$x_p(t)$	Point on path segment closest to $x(t)$

I. Introduction

A large part of the research at the Autonomous Flight Systems Laboratory is directed toward the development of algorithms which govern the actions of a single UAV or a team of UAVs at an abstract level. These algorithms govern high level behaviors such as task and path planning, but assumes that low level algorithms already exist to handle tasks such as state stabilization and signal tracking.

The end goal for many of these algorithms is fully autonomous behavior without input from human operations. However, there are many benefits for allowing human interaction with the system. Verifying and validating a fully autonomous algorithm through an unmanned flight test requires significant logistical

*PhD Candidate, Dept. of Aeronautics and Astronautics, lum@u.washington.edu, AIAA student member
†Research Assistant, Dept. of Aeronautics and Astronautics, rowlandm@u.washington.edu
‡Assistant Professor, Dept. of Aeronautics and Astronautics, rysdyk@aa.washington.edu, AIAA member

planning and development. Many other UAV subsystems that are not directly related to the core strategic algorithm must be developed in order to support the mission. Furthermore, these autonomous flight tests must be conducted in controlled airspaces and under strict supervision. All of these factors greatly increase complexity and development time of the overall system. Many of these problems can be addressed simply by introducing human decision making and interaction at very specific points in the system.

This paper presents a ground based, distributed testing environment which is used at the Autonomous Flight System Laboratory to test high level algorithms by using a human operator in place of several low level systems. In this fashion, the overall system operates in a manner very similar to the fully autonomous system. This system offers many benefits. The main advantage is that these high level algorithms can be implemented and tested much faster and with significantly less effort. In addition, applications can be developed for a standard Windows based environment instead of embedded real time systems. Furthermore, these algorithms can be tested in unrestricted airspaces due to the fact that they are operating as a pilot-assisting system which the pilot is free to ignore at any point.

The main drawback with this architecture is the introduction of a human operator or pilot who may behave in a non-deterministic fashion. To alleviate this problem, the distributed ground based simulator is used to train potential pilots to interact efficiently and consistently with the system.

Many other research laboratories use ground based simulators to verify and validate control algorithms. The UAV Lab at Georgia Tech has developed significant software and hardware[1] for testing algorithms and implementing flight tests.[2] Many of the interfaces and architecture presented in this paper are based on similar ideas. Previous work at the Autonomous Flight Systems Laboratory regarding simulation[3,4] and flight testing[5] of autonomous algorithms has led to the development of the current ground based testing facility. The ground based simulator is easily modified and it being expanded for a recent Multidisciplinary Research Program of the University Research Initiative (MURI) titled Biologically-Inspired Collaboration of Heterogeneous Unmanned Surveillance Systems.

This paper details the control architecture used to integrate human operators with autonomous systems in order to perform verification and validation of high level control algorithms in an efficient manner. The paper also discusses the distributed ground based simulator that is used to simulate actual flight conditions and train human pilots. An example of a pilot interacting with a path planning algorithm[6] is used to illustrate these ideas. Section II introduces the different system architectures which are used in both the ground testing environment and also for flight tests. The various hardware and software used for supporting both the simulator and flight tests are described in Section III. Results with both systems are shown in Section IV before finalizing with some conclusions and lessons learned in Section V.

II. System Architecture

Almost all autonomous algorithms and control laws can be classified by the level of autonomy they achieve. Three common classifications are shown in Figure 1.

Low level control algorithms can be classified as "dynamics and control" algorithms. These algorithms address classical control problems such as state stabilization and signal tracking. These algorithms run at a high bandwidth and are commonly referred to as inner loop or autopilot algorithms. Operating at a slightly higher level are the tactical algorithms. These are more complex algorithms which govern more abstract tasks such as path following and communication/cooperation schemes. Finally, operating at the highest levels of autonomy are the strategic algorithms. These may operate on a much larger time scale and are responsible for mission management types of tasks such as path and task allocation, team management, et cetera.

A. Fully Autonomous Architecture

For a fully autonomous system to operate successfully, control laws must be designed and implemented to address all levels of autonomy. An example of a typical setup for a fully autonomous system is shown in Figure 2.

Low and middle level control algorithms are handled by an inner loop controller and the strategic algorithms are responsible for more abstract, higher level mission management tasks. This type of architecture works well to partition the workload and assign tasks to the controllers. Both the inner and outer loop control laws may be implemented using onboard microprocessors or embedded controllers.[7]

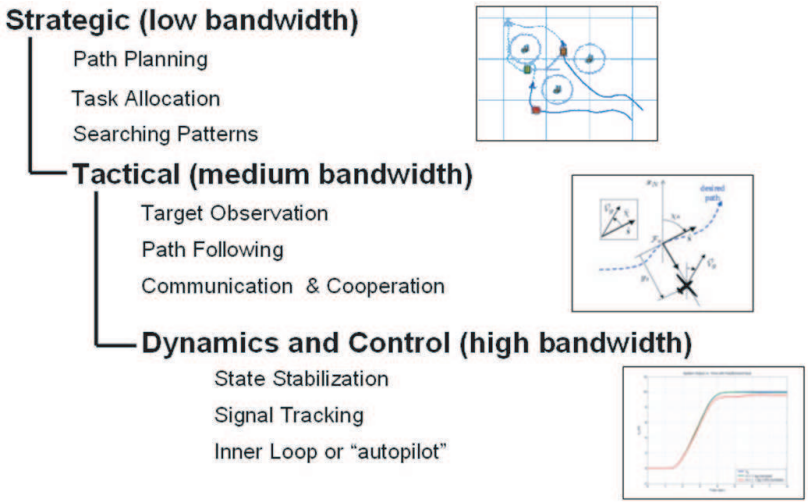


Figure 1. Different levels of autonomy.

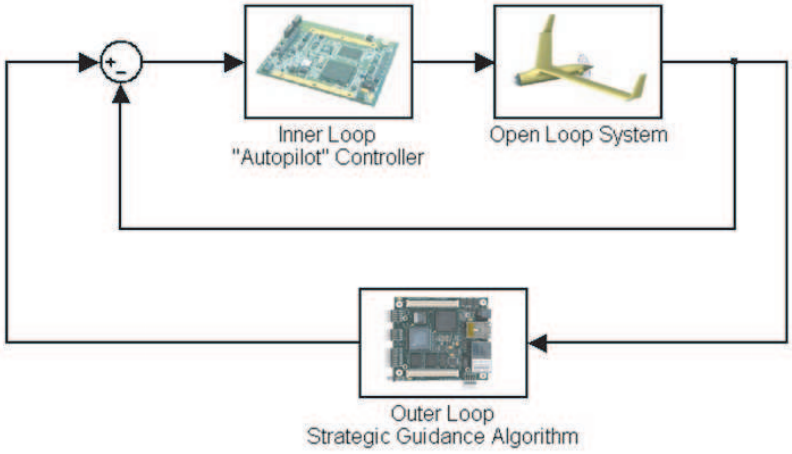


Figure 2. Desired system architecture for fully autonomous flight.

Although this architecture is efficient, the main drawback is due to the fact that although the inner loop control laws can be designed and tested independent from the outer loop controller, the reverse is not true. A valid inner loop must be implemented in order to verify and validate the outer loop. Furthermore, the inner loop is vehicle specific and must be designed and implemented individually for each vehicle. To further complicate matters, often these inner loop applications must be developed to run on embedded controllers, thus requiring specialized training and development environments.

B. Human-in-the-Loop Architecture

Many consider low level control problems such as state stabilization and signal tracking to be mature technologies with limited academic research currently focused on these problems. However, verification and validation of strategic control algorithms are currently an active field of research.

1. Flight Test Architecture

The architecture used at the Autonomous Flight Systems Laboratory for semi-autonomous flight tests replaces the fully autonomous UAV system with a human pilot and flight vehicle is shown in Figure 3.

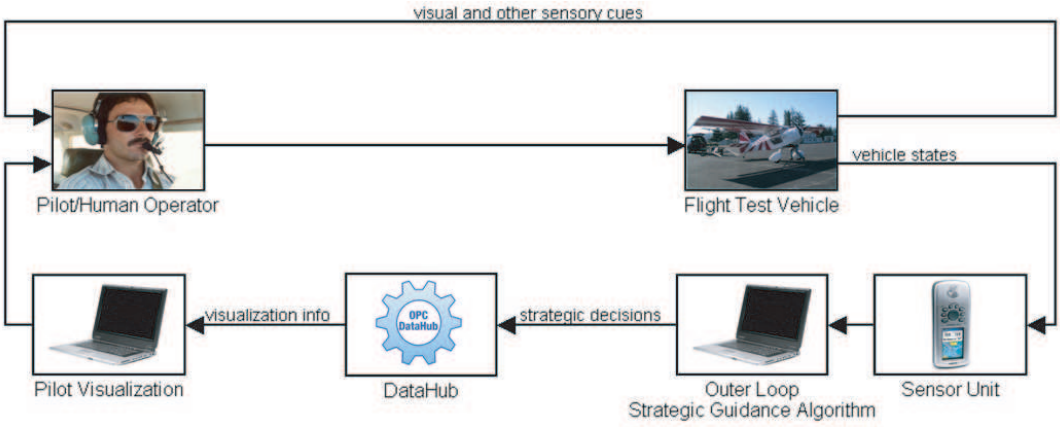


Figure 3. System architecture for human-in-the-loop flight test of strategic controller.

In this setup, the strategic control laws are implemented on a standard laptop PC (see Section III, B). The tasks that are usually handled by the inner loop controller are instead managed by the human pilot. The outer loop relays information to the inner loop (pilot) by displaying pertinent information to a second laptop (called the visualization laptop). The visualization and strategic laptops are connected via software called OPC DataHub (see Section III, A).

There are several advantages to this architecture. The most obvious is that this setup avoids the significant time and effort required to develop a viable inner loop controller for the vehicle. Instead, the human pilot operates as the inner loop controller by taking commands from the outer loop (strategic guidance algorithms). This is similar to the difference between driving a car assisted by GPS navigation and developing a fully autonomous car capable of navigating via GPS.

This setup allows the strategic algorithms to be developed using a standard development environment such as Microsoft Visual Studio or Matlab since the application will be implemented on a laptop PC running a Windows operating system. This further saves time and effort because it is not necessary to port algorithms to an embedded real time operating system (RTOS).

In addition, this architecture allows vehicles using these strategic algorithms to operate in normal airspaces with little or no safety problems (the pilot can choose to ignore commands from the strategic algorithm at any time).

2. Distributed Simulation Architecture

In order to accurately validate the outer loop controller, the inner loop controller must behave in a reliable, deterministic fashion. In other words, the human operator must interface with the system in a predictable manner. One popular method for designing inner loop controllers is to use neural networks.[8] Neural networks have been some of the earliest and most studied models of human brain function.[9] The amount of error introduced by the human operator depends on the amount and type of practice that they receive (i.e. training sets for the network). The system architecture used at the Autonomous Flight Systems Laboratory to train human operators for interfacing with the strategic algorithms is shown in Figure 4.

This setup mirrors the flight test architecture shown previously in Figure 3 except the flight test vehicle and sensor unit are replaced with their simulated counterparts. The vehicle is modeled as a 6 degree of freedom rigid body based on the Research Civil Aircraft Model.[10] The aircraft simulation is implemented on two separate desktop PCs.

III. Software and Hardware

The human-in-the-loop distributed simulator is comprised of several software and hardware systems running in parallel. These systems and their functions are described below.

The physical setup of the distributed human-in-the-loop simulator is shown in Figure 5. In this system,

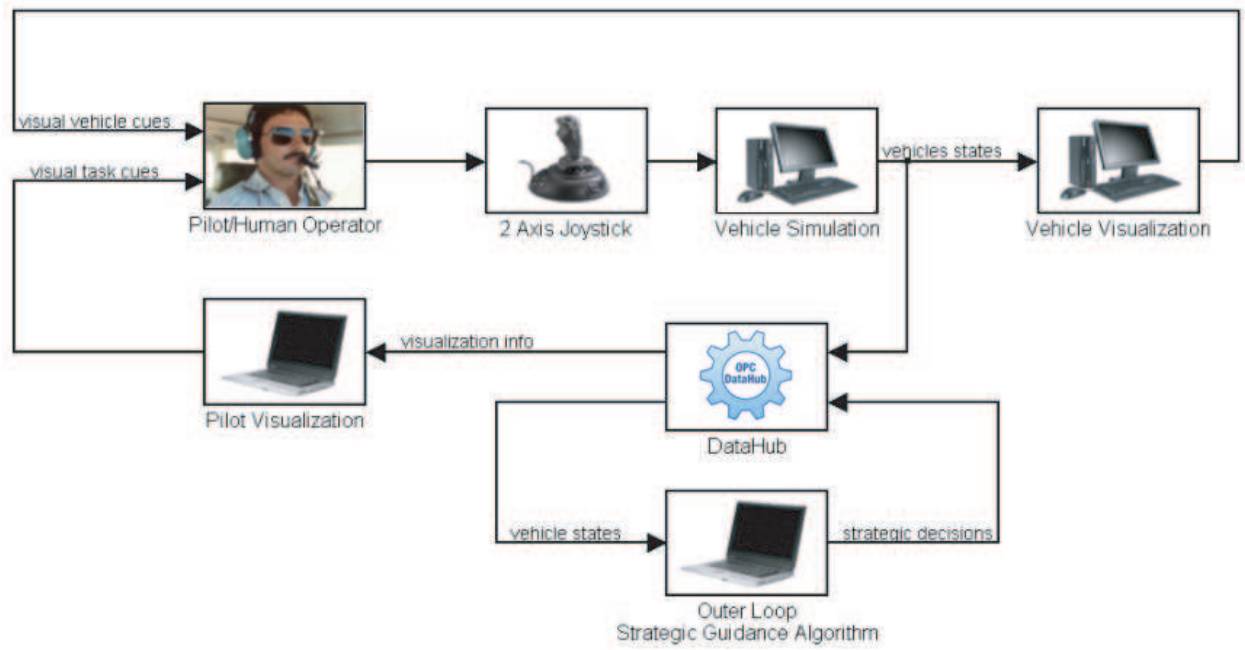
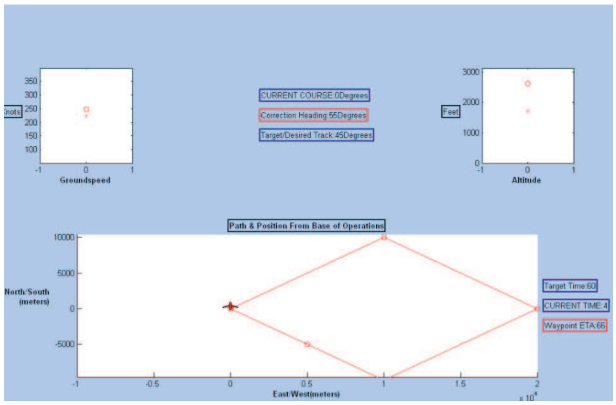


Figure 4. System architecture for ground based distributed human-in-the-loop simulation.

(a) Physical setup



(b) Pilot visualization

Figure 5. Physical setup of Distributed Human-in-the-Loop Simulator and screen shot of operator visualization.

the pilot interacts with the simulator via a three axis joystick and receives visual cues regarding the state of the aircraft from the FlightGear[11] output. Information regarding the output of the strategic algorithm is displayed to the pilot via a dedicated laptop. A screenshot of the information conveyed to the pilot is shown in Figure 5(b). In this situation, the strategic algorithm is a path planner and outputs information regarding the path that the pilot is required to fly.

The output consists of four separate screens. In the upper left, a display showing the desired groundspeed and the current groundspeed is display. In the upper right, a similar display is used for altitude tracking. The display on the bottom consists of a top view of the current path and the location and orientation of the agent. Directly above this wire frame drawing is a series of numbers indicating the current course angle, the desired course angle, and a correction course angle which will put the agent back on track if there is a non-zero cross track error.[12]

A. Software

Software used in the distributed simulator are all implemented on Windows XP machines. The vehicle simulation is performed using Matlab Version 7.2.0.232 (R2006a) and Simulink 6. In addition, the AeroSim blockset version 1.2[13] is used to interface to the two axis joystick with the Simulink model.

To distribute processing power, the vehicle visualization is handled by a separate desktop PC. This PC renders the vehicle in in 3D using FlightGear v0.9.8. The AeroSim blockset is used to interface the Simulink model of the vehicle with this visualization tool.

Another crucial piece of software is the OPC DataHub. This is a centralized database with a publish/subscribe architecture where multiple applications can both write to and read from a centralized server. This software is used to transfer data between different applications. Many applications have tools for interfacing with the DataHub. The OPC Toolbox in Simulink allows signals from the Simulink model to be written to (or read from) the DataHub.

The pilot visualization is also implemented using Matlab and Simulink and uses the OPC Toolbox to read relevant information from the DataHub.

The strategic algorithm is implemented as a standalone executable. In the Autonomous Flight Systems Laboratory, it is developed using the C++ language. This allows separate software objects to be dedicated to interfacing with the sensor and the DataHub.

Software applications used in the system are designed to be modular so that they can be developed and improved independently. Because multiple developers work on different pieces of software at different times, the lab makes use of TortiseSVN to handle version control of code.

B. Hardware

The distributed simulator is made up of several hardware components as well. The main hardware is the series of networked desktop and laptop PCs. The various computers, the applications they host, and their functions are summarized in Table 1.

Table 1. Hardware and software components used by various machines in distributed simulator.

Name	Type	Applications	Function
mica-e1	Desktop PC	Matlab Version 7.2.0.232 (R2006a), Simulink 6 with AeroSim version 1.2 blockset and OPC Toolbox, OPC DataHub version 6.3.14.166	Vehicle state and environment simulation and human interface system
mica-e4	Desktop PC	FlightGear v0.9.8	Vehicle visualization
mica-m1	Laptop PC	Matlab Version 7.2.0.232 (R2006a), Simulink 6 with OPC Toolbox, OPC DataHub version 6.1.9.133	Pilot Visualization and OPC DataHub server
lcc-110	Laptop PC	Strategic Algorithm	Strategic Algorithm carrier and sensor interface

The Desktop PC responsible for the vehicle simulation (mica-e1) is also responsible for interfacing with the human operator. This is done via a three axis Microsoft Sidewinder joystick.

For flight tests, the vehicle and sensor model are replaced with an actual aircraft and sensor. Position is obtained using a Garmin GPSMAP76CSx GPS unit connected to the Strategic Algorithm laptop via a serial cable. The hardware which is carried in the vehicle during flight tests is shown in Figure 6.

IV. Results

A. Path Following Example

The distributed simulator was developed in order to verify and validate a strategic team management algorithm.[3] Previous research at the Autonomous Flight System Laboratory has been focused on developing

Figure 6. Flight test hardware.

strategic path planning algorithms[14].[15] These strategic algorithms generate paths for the agent to follow. For this example, the distributed simulator is used to train pilots to perform a path following task.

A feasible path for the agent consists of a sequence of waypoints ($p_i \in \Re^3$) where each consecutive waypoint is no more than a distance r_{max} away from the previous one. In addition, each of these waypoints specifies a time, t_i when the agent must arrive at the location p_i. An example of such a path is shown in Figure 7. In this example, the algorithm plans a path (shown in red) for the vehicle to navigate through the environment.[14] Once the path is determined and displayed to the pilot, it now becomes the job of the pilot to follow this path to the best of their ability.

The skill and ability of the pilot is measured using a performance metric. The position of the simulated agent is recorded every ΔT seconds and is denoted $x(t)$. At each time t, the agent should ideally be located on the line segment joining waypoints p_i and p_{i+1} for some $i \in \{0, 1, \ldots, d-1\}$ (for the case of $i = 0$, $p_0 = x(0)$). For convenience, the point p_i is referred to as A (the starting waypoint) and the point p_{i+1} is referred to as B (the ending waypoint). A and B have respective time stamps t_i and t_{i+1}. If the agent is not on this line segment, the agent is off the path and the point $x_p(t)$ can be found which is the point on the line segment with minimum Euclidean distance from the agent's current location, $x(t)$. The point $x_p(t)$ at each time step is given by

$$x_p(t) = A + \alpha^\star(B - A) \tag{1}$$

In this situation, $\alpha^\star$ is a scalar in the range of $[0, 1]$ which denotes how far from A to B the point $x_p(t)$ is. It is obtained by solving a minimization problem of $\alpha^\star \in \arg\min f_0(\alpha, t) = \frac{1}{2}||x(t) - [A + \alpha(B - A)]||^2$

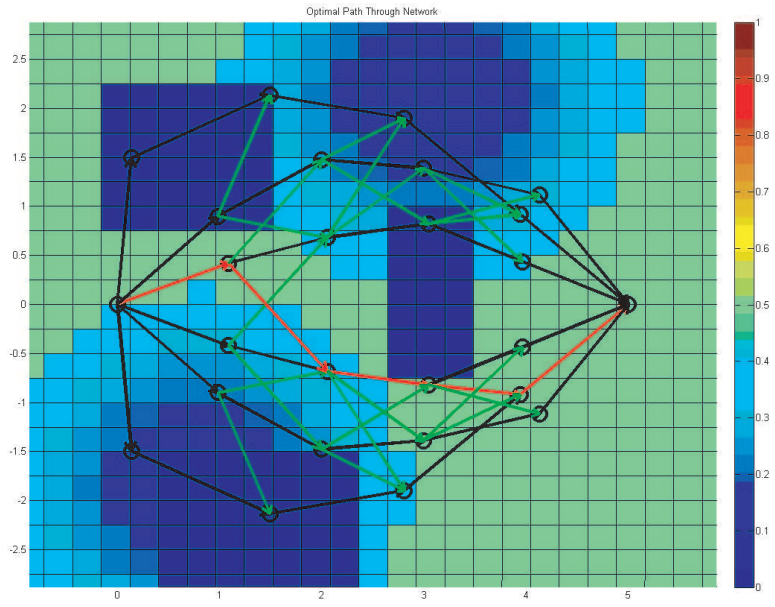


Figure 7. Example of a path generated by strategic algorithm shown in red.

over all $\alpha \in [0, 1]$. The solution can be analytically found to be

$$\alpha^\star = \begin{cases} 1 & \text{if } \beta > 1 \\ 0 & \text{if } \beta < 0 \\ \beta & \text{otherwise} \end{cases} \tag{2}$$

$$\text{where } \beta = \frac{A^T(A-B)+(B^T-A^T)}{A^TA-2A^TB+B^TB}x(t)$$

So the instantaneous cost at time step t is given by

$$f_0(\alpha^\star, t) = \frac{1}{2}||x(t) - [A + \alpha^\star(B - A)]||^2 \tag{3}$$

The accumulated cost up to time t is simply the sum of all the instantaneous costs up to the current time

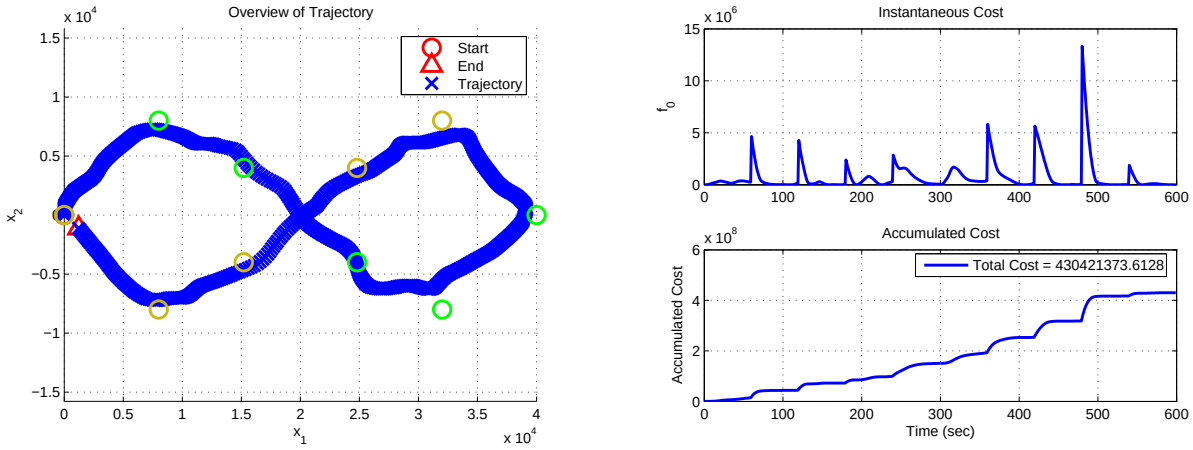
$$J(t) = \sum_{\tau=0}^{\tau=t} f_0(\alpha^\star, \tau) \tag{4}$$

An example of an unskilled pilot flying through two paths and the instantaneous and accumulated cost is shown in Figure 8.

In this situation, the pilot is tasked with flying a figure eight pattern. The pattern is broken up into two distinct paths. The first path consists of five green waypoints (p_i for $i = 1, ..., 5$), each of which have desired arrival times of t =60, 120, 180, 240, and 300 seconds, respectively. At $t = 300$, a second path is generated which consists of the five brown waypoints which return the agent to the starting point.

The pilot's performance is measured and displayed in Figure 8(b). The instantaneous cost is a measure of how far off the desired path the pilot is. Notice that the discontinuities in instantaneous cost occur when the active waypoint changes from p_i to p_{i+1} (for path 1, this occurs at t =60, 120, 180, 240, and 300 seconds). These jumps are due to the fact that at time $t < t_i$, the agent has not reached waypoint p_i but is roughly on the desired path between p_{i-1} and p_i. This results in a low cost (most likely that $\alpha^\star \in [0, 1]$). However at $t > t_i$, the next active waypoint becomes p_{i+1} and since the agent has not yet reached point p_i, the cost becomes large (most likely that $\alpha^\star = 0$).

The performance of the pilot can be judged by the instantaneous cost trace. Skilled pilots will have a low average value with minimal discontinuities. For identical paths and times, the performance can also be judge by $J(t_f)$ which provides a type of score for the run.



(a) Overview of trajectory

(b) Instantaneous and accumulated cost

Figure 8. Simulator results from human-in-the-loop simulation.

V. Conclusion and Further Research

This paper presented an architecture for verifying and validating the performance and output of strategic control algorithms with a high degree of accuracy while minimizing time from development to flight test. This was done by introducing human interaction at specific points in the system which preserves the autonomous contributions of the strategic algorithm by reducing the human to a simple inner loop controller. This architecture is used in the distributed ground based simulator to simulate flight test conditions in a controlled environment. The ground based distributed simulator is used to verify and validate the strategic control algorithms and also to familiarize pilots with the interface and to train them before actually performing a flight test.

During the development of the distributed simulator, many different ways of allowing human interaction were experimented with. Significant development effort was focused on developing a pilot visualization system which relays the pertinent path information to the pilot. By interviewing pilots after using they had used the system, it was discovered that most of them only rely on the overview of the path (the bottom graph in Figure 5(b)). The operator workload became too high when they were forced to constantly scan all four displays and process the information. Current research is directed towards developing a single display which efficiently conveys all the relevant information to the pilot.

The other major interface between the simulator and the pilot involves the joystick. The current joystick is only a three axis device. The two principal axes were mapped to elevator and aileron inputs. Initially, the third axis was mapped to the throttle input and the rudder was fixed to zero. This was designed because many commercial pilots expressed that rudder input is typically reserved for a yaw damping system. However, it was discovered that path following was easier for the pilot if sideslipping and coordinated turns were allowed. To facilitate these maneuvers, the third axis was mapped to be the rudder input. The throttle was instead controlled using the buttons and an integrator scheme.

Finally, the current joystick automatically re-centers the two main axes to zero when the pilot takes their hands off the stick. Many pilots were displeased by this because it made it difficult to trim the aircraft during a run. The next generation simulator will include a more sophisticated joystick with more than three axes and trim features.

VI. Acknowledgements

This work is sponsored in part by the Washington Technology Center (WTC) under grants F04-MC2 and F05-MC3 and the Osberg Family Trust Fellowship. The authors would also like to thank other members of the Autonomous Flight Systems Laboratory, Anawat Pongpunwattana and Richard Wise for contributions to the simulator.

References

[1]Johnson, E. N., Rooz, N., Hur, J., and Pickell, W., "A Concurrent Testing Process for Research Unmanned Aerrial Vehicles," *Proceedings of the 25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, San Francisco, CA, June 2006.

[2]Proctor, A. A., Kanna, S. K., Raabe, C., B., C. H., and Johnson, E. N., "Development of an Autonomoous Aerial Reconnaissance System at Georgia Tech," *Proceedings of thesociation of Unmanned Vehicle Systems International Unmanned Systems Symposium and Exhibition*, 2002.

[3]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Occupancy Based Map Searching Using Heterogeneous Teams of Autonomous Vehicles," *Proceedings of the 2006 Guidance, Navigation, and Control Conference*, Autonomous Flight Systems Laboratory, Keystone, CO, August 2006.

[4]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Autonomous Airborne Geomagnetic Surveying and Target Identification," *Proceedings of the 2005 Infotech@Aerospace Conference*, Autonomous Flight Systems Laboratory, Arlington, VA, September 2005.

[5]Pongpunwattana, A., Wise, R., Rysdyk, R. T., and Kang, A. J., "Multi-Vehicle Cooperative Control Flight Test," *Proceedings of the 25th Digital Avionics Systems Conference*, October 2006.

[6]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.

[7]Anderson, D. E. and Pita, A. C., "Geophysical Surveying with GeoRanger UAV," *Proceedings of the 2005 Infotech@Aerospace Conference*, The Insitu Group, Arlington, VA, September 2005.

[8]Kim, B. S. and Calise, A. J., "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control and Dynamics*, January 1997, pp. 26–33.

[9]Borrett, D., Kelly, S., and Kwan, H., "Phenomenology, Dynamical Neural Networks and Brain Function," *Philosophical Psychology, Vol. 13, No. 2*, 2000, pp. 213–228.

[10]Lambrechts, P., Bennani, S., Looye, G., and Helmersson, A., "Robust Flight Control Design Challenge Problem Formulation and Manual: the Reserach Civil Aircraft Model (RCAM)," Tech. rep., Group or Aeronautical Research and Technology in Europe, Europe, 1997.

[11]"FlightGear Flight Simulator," Public Information, http://www.flightgear.org/.

[12]Rysdyk, R. T., "Unmanned Aerial Vehicle Path Following for Target Observation in Wind," *Journal of Guidance, Control, and Dynamics*, September 2006, pp. 1092–1100.

[13]Unmanned Dynamics, Hood River, OR, *AeroSim Aeronautical Simulation Blockset User's Guide Version 1.2*.

[14]Lum, C. W. and Rysdyk, R. T., "Time Constrained Randomized Path Planning Using Spatial Networks," Tech. rep., University of Washington, Seattle, WA, 2007.

[15]Pongpunwattana, A. and Rysdyk, R. T., "Real-Time Planning for Multiple Autonomous Vehicles in Dynamic Uncertain Environments," *AIAA Journal of Aerospace Computing, Information, and Communication*, December 2004, pp. 580–604.