AIAA Guidance, Navigation, and Control Conference and Exhibit
16 - 19 August 2004, Providence, Rhode Island

AIAA 2004-5142

Implementation of a Manned Vehicle - UAV Mission System

M. Valenti*, T. Schouwenaars†, Y. Kuwata‡, E. Feron§ and J. How¶
Massachusetts Institute of Technology, Cambridge, MA 02139

J. Paunicka‖
The Boeing Company, St. Louis, MO 63166

We discuss the development, integration, simulation, and flight test of a manned vehicle - UAV mission system in a partially-known environment. The full control system allows a manned aircraft to issue mission level commands to an autonomous aircraft in real-time. This system includes a Natural Language (NL) Interface to allow the manned and unmanned vehicle to communicate in languages understood by both agents. The unmanned vehicle implements a dynamic mission plan determined by a weapons systems officer (WSO) on the manned vehicle. The unmanned vehicle uses a Mixed-Integer Linear Programming (MILP) based trajectory planner to direct the vehicle according to the mission plan. We provide simulation and test results for this system using an integrated computer-based simulation and the vehicle hardware testing in late June 2004. The activities described in this paper are part of the Capstone demonstration of the DARPA-sponsored Software Enabled Control effort.

I. Introduction

Unmanned aerial vehicles (UAVs) are used by both military and civilian organizations in a number of applications. Recent advances in guidance technologies have enabled some UAVs to execute simple mission tasks without human interaction. Many of these tasks are pre-planned using reconnaissance or environment information. For example, air operations are executed according to an Air Tasking Order (ATO), which may take up to 72 hours to plan, task and execute.[1] In volatile situations, information about the vehicle's operating environment may be limited. Therefore, task planning flexibility and safe trajectory solutions are essential to the survivability and success of an autonomous system.

Although guidance systems have existed for many years, most unmanned vehicles do not exhibit the level of performance and flexibility needed to complete an entire mission autonomously. More specifically, unexpected changes in the environment may require changes to its mission-level plan. In this case, the vehicle's guidance and mission planning systems must possess enough intelligence (and processing power) to recognize and react to changes in the operating conditions. On the other hand, if the vehicle is ordered to perform a new mission task during an operation, it must possess the capability to safely transition from its previous trajectory to its new mission goals.

*Student Member AIAA, Ph.D. candidate, Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 valenti@mit.edu

†Student Member AIAA, Ph.D. candidate, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 toms@mit.edu

‡Student Member AIAA, Ph.D. candidate, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 kuwata@mit.edu

§Associate Fellow AIAA, Associate Professor of Aeronautics and Astronautics, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 77 Massachusetts Ave, M.I.T. Rm 32-D724, Cambridge, MA 02139 feron@mit.edu Author to whom all correspondence should be addressed

¶Senior Member AIAA, Associate Professor of Aeronautics and Astronautics, Aerospace Controls Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 jhow@mit.edu

‖Associate Technical Fellow, Boeing Phantom Works, The Boeing Company, St. Louis, MO 63166 james.l.paunicka@boeing.com

The complexity of this problem increases when a second agent is introduced. For example, if another autonomous agent is added to the mission scenario, then both vehicles must resolve information regarding the impending actions of the other vehicle. Similarly, if a manned agent is added to the original scenario, the autonomous vehicle must also possess the capability to effectively communicate and coordinate its actions to the manned vehicle.

This paper discusses the development, implementation, and evaluation of a system containing a manned vehicle and a UAV mission system in a partially-known environment. This system provides a manned aircraft with the ability to issue tasks and mission-level commands to an unmanned aircraft in real-time. The activities described in this paper are part of the Capstone demonstration of the DARPA-sponsored Software Enabled Control (SEC) effort. This system was developed with a Natural Language Interface, which interprets sentence commands from the manned vehicle operators to a set of codes understood by the unmanned vehicle, and vice versa. In addition, the system uses a Mixed-Integer Linear Programming (MILP) based trajectory planner integrated with a task scheduler to implement a dynamic mission plan provided by the weapons systems officer (WSO) in the manned vehicle to the unmanned vehicle in real-time. The MILP based guidance algorithms optimize the vehicle's flight path through the partially-known environment according to the mission plan. Finally, we will briefly discuss the test results from real-time software-in-the-loop (SIL) simulations, hardware-in-the-loop (HIL) simulations and test flights performed at Edwards Air Force Base and NASA Dryden Flight Test Facilities. These flight tests mark the first time that a Natural Language Interface has been used by a manned vehicle to task a UAV in real-time. They also mark the first time that a MILP based Guidance System has been used to control a UAV in coordination with a manned vehicle.

II. Background

As part of the DARPA-sponsored Software Enabled Control program, one of our team's goals was to develop guidance and mission technologies for unmanned vehicle systems. Much of our team's research effort focused on developing technologies to confront issues facing systems with multiple unmanned vehicles, such as safety, vehicle control, mission/task communication, and multiple vehicle coordination. Most of the development of these technologies was completed in local testing environments using computer-based simulations.

To conclude this program, a Capstone technology flight test demonstration was scheduled. This demonstration offered our team the chance to transition some of our mission and guidance technologies to a flight-test platform. In this demonstration, our team was provided access to three types of vehicles: a Manned Fixed-Wing (FW) vehicle, an Unmanned Aerial Vehicle (UAV), and a Ground Station. The FW vehicle was to be flown by a pilot, the UAV was to be directed by our technology, and the ground station would monitor the experiments.

Based on these assets, it was clear that there were a number of challenges to overcome in developing, implementing and demonstrating our technology on these flight-test platforms. For example, the UAV guidance technology must be robust to changes and threats in the vehicle's environment - including obstacles, changing wind conditions and other vehicles. Second, if the FW vehicle and UAV were going to coordinate their activities during the mission, a mechanism allowing both vehicles to communicate with one another needed to be developed. Thirdly, since this mission was taking place in real-time, the technologies developed for the experiment had to reach a solution and resolve any unexpected issues reliably in a pre-defined period of time.

III. Technology Development

From these challenges, our team decided to focus on a scenario where the UAV flew a mission in support of the FW vehicle. This type of mission requires the UAV perform reconnaissance tasks for the FW vehicle on a multiple vehicle mission. In addition, since the FW vehicle also has mission goals, the pilot and/or weapons systems officer (WSO) aboard the FW vehicle will not have the time to direct the UAV using low-level guidance commands (e.g., "Turn left", "Speed up") without sacrificing some of their own capabilities during the mission.

Therefore, this scenario places an emphasis on the autonomous mission capabilities of the UAV. As a result, a reliable and easy-to-use interface between the manned FW vehicle and UAV was needed to simplify

the communication of information between both vehicles in a concise manner. Also, a mission system on-board the UAV was needed to interpret and translate commands from the FW vehicle into safe, achievable trajectories, while coordinating the UAV's activities with the mission goals of the FW vehicle.

In addition to these mission capabilities, this demonstration required that these technologies could be implemented in a real-time environment. Some of these real-time challenges included:

- *Memory and resource allocation* Since this system runs in real-time, the guidance system must make provisions for a safe and achievable solution in a pre-defined period of time using the hardware provided.
- *Communication and Bandwidth Issues* The Link-16 communications system sends information every second. As a result, there was an upper limit to the amount of user information that can be transmitted at every iteration. Therefore, the user information transmitted between both aircraft must be concise and include the necessary information required for both vehicles (and their operators) to make decisions.
- *Robustness Issues* Because the environment was not controlled and only partially known, the systems designed for each aircraft had to be robust to external and unexpected conditions. For example, if information between vehicles is not received, either vehicle should re-transmit its data. Likewise, the guidance system must plan paths that are robust to environmental conditions (i.e. pop-up obstacles, no-fly zones, changes to the mission plan, wind conditions).
- *Software Integration Issues* Since each software module for the demonstration was in development at the same time, modifications to any system component could have major effects on the development of the others.

To aid in the transition to a real-time flight test platform, our demonstration software was to be integrated with the Boeing Open-Control Platform (OCP) software and loaded onto a laptop fitted in each aircraft. The Boeing OCP software[2] provided our team with an aircraft interface to some of (but not limited to) the following abilities:

- Send and receive state and user-defined data between both aircraft using a Link-16 Communications interface
- Receive the current vehicle state data
- Send a set of pre-defined commands to the aircraft avionics system which include (but not limited to) Set and Hold Turn Rate, Set and Hold Speed, Set and Hold Altitude, Set and Hold Heading, etc.
- Memory storage and Time Frame Execution

Using the Boeing OCP utilities, our demonstration software was designed to use a subset of these resources.

To address the requirements of this mission scenario, our team focused on three major technology areas:

1: *Natural Language Interface*
This component interprets and converts normal sentence commands (e.g., "Search this region for threats") from the humans on-board the FW vehicle into data the UAV can use and understand and vice-versa.

2: *Task Scheduling and Communication Interface*
The primary goal of this component is to interpret the command data from the Natural Language interface component and develop a series of tasks the vehicle can perform.

3: *Trajectory Generation Algorithms*
This component uses information about the vehicle's environment and mission goals to generate trajectory and vehicle commands to guide the UAV throughout the mission.

Each of these components address a capability required to perform this mission. These three components were developed simultaneously for this effort and were integrated in early December 2003 for system level testing. In the following sections, we discuss the development of these three technologies and how they are integrated into the demonstration system.

IV. Natural Language Parser and Interface

As part of this mission, the pilot and/or weapon system officer require the capability to interact with the UAV. As such, the goal of any such system must be to minimize the workload on the operator, while efficiently and effectively communicating with the UAV. This is one of the large benefits in using a Natural Language Interface: it allows a human operator to interact with a computer-based system using the human's natural language.

Therefore, the Natural Language Interface module was designed to use a natural language parser that provided an operator with the ability to communicate with the UAV using normal sentence commands (in English for this demonstration). The key feature of this system is that the natural language commands from a human are translated into an optimization problem which will guide the UAV. For example, if the FW operators are notified of a potential threat, the FW operators could command the UAV to search a pre-defined region containing the potential by using the following sample dialog:

FW: "UAV 7, this is Eagle 3."

UAV: "Go ahead, Eagle 3."

FW: "Add New Mission Task. Proceed to location Echo-Charlie 5 in Minimum Fuel. Search this region for threats and wait for further instructions after the task is completed"

UAV: "Roger. Acknowledge task information - proceeding to location Echo-Charlie 5."

FW: "Eagle 3, out."

In this dialog, there are four major components to the conversation:

1: *Introduction* The object beginning the conversation (the FW WSO in this situation) identifies the object who is to receive and react to this message (which is the UAV in this case).

2: *Response / Verification* The object receiving the introduction message or command responds with a verification that it received and understood the incoming message.

3: *Command / Question* This message contains the "information-carrying" portion of the conversation and requires a detailed response from the receiving party.

4: *Closure* This message signals the end of the conversation. In our scenario, the FW (which represents the commanding officer) will always end the conversation with "*(FW WSO Identifier)*, out." for a definitive end to the conversation.

In the conversation above, the command and response portions require the vehicle to send a message with information about the current status of the mission, task, etc. Therefore, we developed a protocol for questions, commands and their associated responses – for both sentences (for the human user) and their coded representations (for the machine) – to ensure that the messages communicated between both agents are accurate representations of the conversation. For this particular application, we observed that the introduction and closure statements are used to identify the parties involved in the conversation. Therefore, since there are only two parties (the FW WSO and UAV) involved in this demonstration, the system will not send these messages over the communications link to simplify the communications protocol for this system.

V. Task Scheduling and Communications Interfacing

The task scheduling and communications processing module components are designed to centralize all of the UAVs Mission Processing in one module. Therefore, if the UAV is commanded to perform a task by the FW vehicle, the UAV would keep track of the mission tasks, waypoint locations and known obstacle list to pass on to the navigation algorithms (in a pre-defined form) as requested. In this demonstration, both the mission task planner and the navigation algorithms run at 1 Hz, so it was natural to develop the system to run its mission pre-processing before the trajectory generation phase.

For this demonstration system, the communications processing module provides the FW WSO with the authority to send task commands and receive status updates, threat/obstacle avoidance information

and acknowledgement information, while providing the operators monitoring the UAV with the ability to override the navigation system in the event of an emergency or error. The system sends threat and override information to the FW WSO before any status/update information in an effort to send the most important data relevant to the demonstration before any auxiliary information. Input/Output data is processed every 1 Hz frame before the task planner and pre-processing step to ensure that the most up-to-date information is used by the UAV trajectory planner.

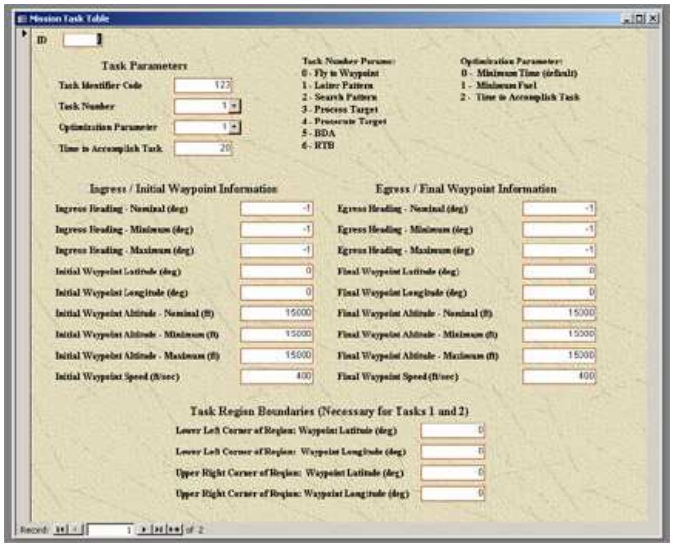


Figure 1. Pre-Defined Mission Task Library Database

Currently, the task scheduling module allows a user to plan a number of tasks using a pre-defined list or as programmed during a mission. Since many missions are pre-planned, the system allows a user to initiate a pre-defined mission task (created using the User-Defined Mission Task Library Database, as shown in Figure 1) or modify/create a mission plan by entering the parameters in the list described below. The current list of mission tasks include:

- Fly to Waypoint X
- Loiter Pattern
- Search Pattern
- Classify Target
- Attack Target
- Battle Damage Assessment (BDA)
- Return-to-Base

For each of these task options, a user must provide the task area ingress / egress conditions, size and location of the task area (given by the lower left and upper right coordinates of the task area). In addition, a user (via the Natural Language interface in real-time) has the option of providing the optimization parameter used by the trajectory generation algorithm (minimum time, minimum fuel, or the amount of time to finish the task). Next, the user must decide if they want to give the vehicle a new task to perform or change the current task the vehicle is performing. The "New Task" command is added to the end of the UAV task list and will be performed after all of the tasks current in the scheduler have been completed. The "Change Task" command modifies the current task performed by the UAV. Once a task is completed by the UAV, it is removed from the UAV task list. To reduce the complexity of this demonstration system, only the current task can be modified – although future versions of this system will have the capability to change any of the tasks in this scheduler.

Finally, for the purposes of the flight demonstration, the FW WSO did not have the capability to define a new task or adjust parameters in the current task manually because of communications link bandwidth constraints. Instead, the FW WSO was able to command the vehicle to perform tasks from the pre-defined task library using the experiment keys on the FW Laptop. When a task is selected, the UAV automatically updates the mission task list and sends an acknowledgement to the FW WSO. The updated task information is included in the data sent to the trajectory planning algorithms.

VI. Trajectory Planning Algorithms

After the mission processing steps have been converted into a series of tasks for the vehicle to perform, the trajectory generation module is needed to guide the vehicle from one location to another given information about the current state of the vehicle and the environment. In this scenario we assume that the environment is partially-known and explored in real-time. Therefore, the guidance algorithms use a receding horizon planning strategy, which accounts for new information about the environment at each time step.

Both trajectory planning algorithms implemented for this demonstration use Mixed-Integer Linear Programming (MILP), a powerful optimization framework that extends continuous linear programming to include binary or integer decision variables.[3] These variables can be used to model logical constraints such as obstacle and collision avoidance rules, while the dynamic and kinematic properties of the aircraft are formulated as continuous constraints.[4]

We have developed two different guidance algorithms for this demonstration, as described below. Both use a closed loop state space model $(\mathbf{A}, \mathbf{B})$ of the aircraft in an inertial reference frame, resulting from augmenting the vehicle with a waypoint tracking controller.

1. "Safe Trajectory" Algorithm

The first algorithm is designed to promote safe trajectory planning for a specified distance around a vehicle in a partially-known environment. Safety is ensured at all time by constraining the trajectories computed at each time step to terminate in loiter circles that do not intersect with the known no-fly zones and other obstacles. The loiter circle should lie within the detection radius of the aircraft, *i.e.*, it should be placed in a region of the environment that is fully characterized at the current time step. Assuming that the planned trajectories can be accurately tracked, at each time step, the remaining part of the previous plan together with its loiter circle can serve as a safe backup or "rescue" plan. Namely, if at a certain time step, the aircraft fails to find a feasible solution to the MILP problem within the timing constraints of the real-time guidance system, it can just keep following the previous trajectory. Eventually, that trajectory will end in a loiter circle in which the aircraft can safely remain for an indefinite period of time, *i.e.*, without flying into an obstacle or no-fly zone.

In Ref. 5, linear geometric expressions for points along a circle are derived as a function of the last state in the horizon, which serves as the ingress state of the loiter. The circle can be either left or right turning and its radius scales linearly with the magnitude of the ingress velocity. The choice of turning direction and the scaling property give the aircraft more degrees of freedom to fit its loiter circles in the obstacle-free areas of the environment.

To implement this algorithm, the task scheduling pre-processing step included software that calculated intermediate waypoints en route to the desired destination. This software was used to, first, develop a truncated obstacle list which included only obstacles in the known-environment based on the vehicle's location. Next, this software included a search algorithm to determine the vehicle's next intermediate waypoint (based on the vehicle's velocity and the necessary safety distances pre-calculated by the "Safe Trajectory" algorithm) in two steps. First, it determined if the desired waypoint destination provided by the task planning software was in the Line-of-Sight of the vehicle. If so, this destination was provided to the "Safe Trajectory" algorithm as the next destination for the vehicle. Otherwise, the algorithm used a two-step search for a safe intermediate destination for the vehicle using the line-of-sight distances between corners of the obstacles in the known-environment. This search algorithm helped to prevent the guidance algorithm from directing the vehicle into "local minima" of the cost function. Finally, the data pre-processing step concluded by re-scaling all of the state and environment data after the next waypoint was determined to decrease optimizer run-time.

After the optimizer calculated a solution, a post-processing step interpreted the data from the optimizer and determined whether the optimizer was able to calculate a safe, dynamically feasible trajectory solution.

If so, the post-processing algorithms developed and saved a new waypoint plan for the vehicle based on the current environment. Otherwise, the vehicle followed the previous waypoint solution calculated in a previous time step.

2. "Cost Map" Algorithm

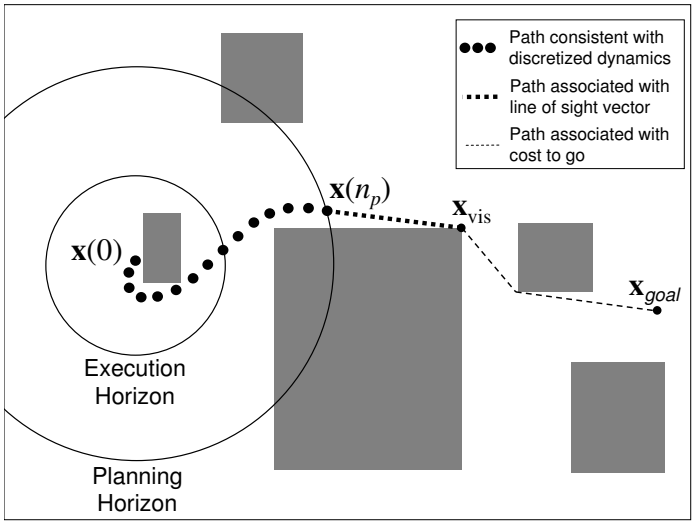


Figure 2. Line-of-sight vector and cost-to-go[6]

The second algorithm, known as the "Cost Map" algorithm, designs a minimum-time path to a fixed goal while avoiding a set of obstacles.[7] Figure 2 gives an overview of the method, including the different levels of resolution involved. The control strategy is comprised of two phases: cost estimation and trajectory design.[8] The cost estimation phase provides the cost-to-go from each obstacle corner by finding visibility graphs and running Dijkstra's algorithm. It produces a *tree* of optimal paths from the goal to each corner. This approximate cost is based on the observation that optimal paths (*i.e.,* minimum distance) tend to follow the edges and corners of the obstacles. In the trajectory design phase, MILP optimizations are solved to design a series of short trajectory segments over a planning horizon. In Figure 2, this section of the plan is shown by the thick dotted line. Each optimization finds a control sequence over n_p steps, but only the first $n_e (\leq n_p)$ control inputs are executed. The margin $(n_p - n_e)$ ensures the vehicle stays in the region from which a feasible path to the goal exists.[9] The vehicle is modeled as a point mass moving in 2-D free-space with limited speed and acceleration to form a good approximate model for limited turn-rate vehicles.

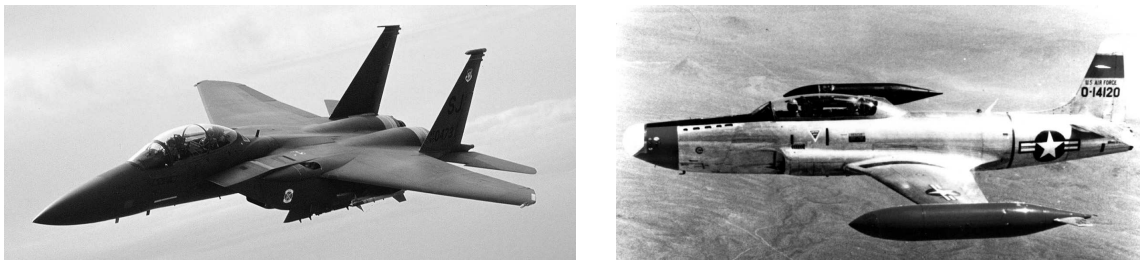
The MILP also chooses a *visible point* $\vec{x}_{vis}$ which is visible from the *terminal point* $\vec{x}(n_p)$ from which the cost-to-go has been estimated in the previous phase. Note that the cost map is quite sparse: cost-to-go values are only known for the corners of the obstacles, but this still provides the trajectory optimization with the flexibility to choose between various routes around the obstacles.

The "Cost Map" algorithm also uses the commercial CPLEX Optimization Solver with Concert Technologies developed by ILOG[10] in calculating a guidance solution. The "Cost Map" algorithm provides a waypoint plan to the post-processing algorithm which invokes a waypoint follower to calculate velocity and turn rate vehicle commands used to guide the UAV through the demonstration area.

VII. Flight Test Scenario Planning and Demonstration Overview

As originally discussed in Ref. 11, our team was tasked with developing a set of flight-test experiments that exhibit the technology we developed. As mentioned before, our team had access to two flight assets for the demonstration:

- Boeing F-15E (*Fixed-Winged aircraft (FW)*)
- Lockheed T-33 (*Unmanned Air Vehicle (UAV)*)



(a) Boeing F-15E Strike Eagle (b) Lockheed T-33 Shooting Star

Figure 3. SEC Capstone Demonstration Test Vehicles

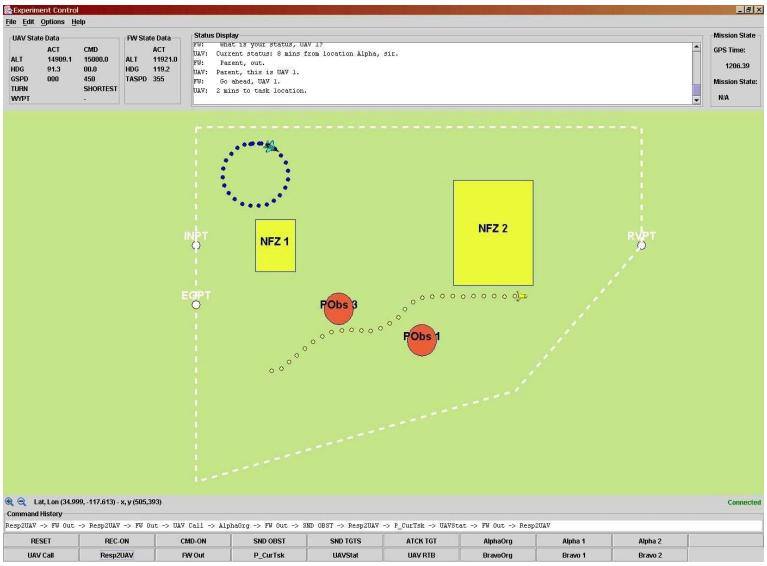


Figure 4. The Boeing Experiment Controller Operator Interface for MIT SEC Capstone Demonstration

A Boeing F-15E fighter jet (similar to the aircraft shown in Figure 3) was piloted throughout the demonstration. This vehicle commanded the UAV to search a partially-known environment for potential targets and threats. The FW WSO communicated with the UAV using a Natural Language (NL) Interface through Boeing's Experiment Controller application, as shown in Figure 4. This interface allowed the FW WSO to communicate with the UAV using sentence commands (in English) throughout the experiment.

A Lockheed T-33 trainer fighter jet (similar to the aircraft shown in Figure 3) was flown to the demonstration area by its two person crew and piloted by a UAV avionics package during the demonstration. The main role of the T-33's two person crew in the MIT experiment was to fly the vehicle to the demonstration area, activate our demonstration software and manage the vehicle in the event of failures during the test flight. The UAV avionics package was interfaced with an on-board laptop running our demonstration software. This software interacted with the UAV avionics package through a set of pre-defined command variables.

In addition, researchers were able to observe the progress of the demonstration from the ground station. This station received state and user-defined information from both vehicles during the flight experiments. During the MIT flight test demonstration, researchers at the ground station were able to communicate with the FW WSO and T-33 GIB and decide which pop-up obstacles and mission task areas were used. The possible locations of the pop-up obstacles and mission task areas were pre-defined, but selected randomly in real-time during each flight experiment.

Based on these demonstration assets, we developed the following mission experiment (shown in Figure 5) to highlight the real-time capabilities and flexibility of our mission software:

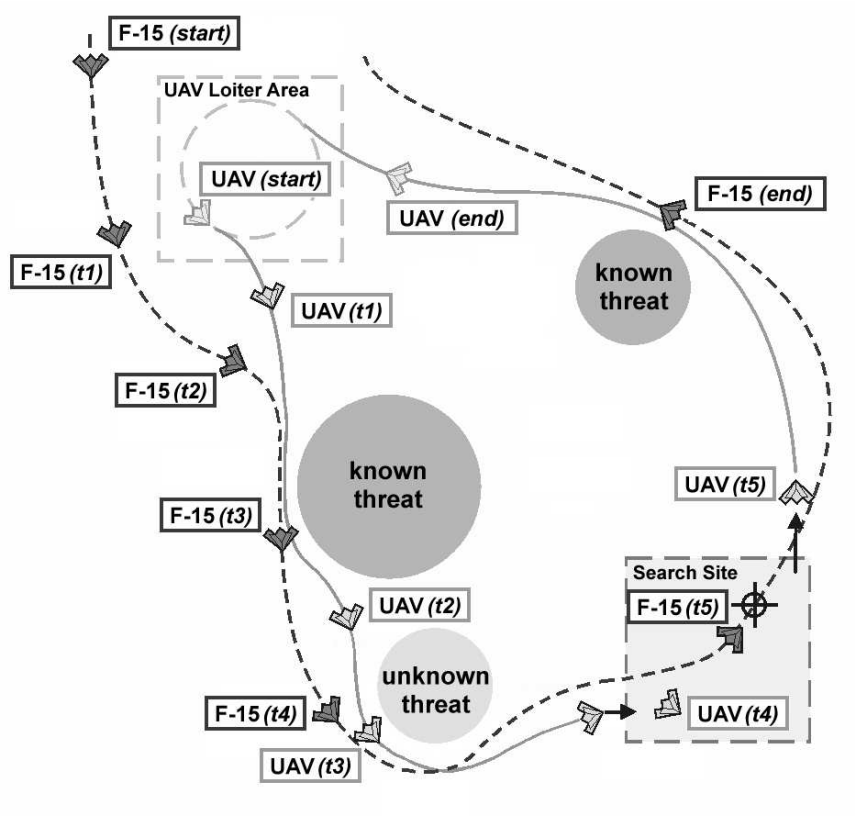


Figure 5. MIT Flight Experiment Mission Overview

Mission: A manned fighter aircraft (FW) and a UAV will work together on a mission to collect images of a possible site in enemy territory. The UAV will perform the reconnaissance for the mission, and the FW operators will decide how the UAV will be used to accomplish the mission goals. The UAV will possess the ability to detect threats and collect images, while the FW vehicle will be able to deliver weapons (if applicable). Since the environment is only partially known, there may be threats to both the manned and unmanned aircraft.

Starting condition: The UAV will start in a pre-defined loiter pattern, and the FW vehicle will be flying an air-patrol near the enemy territory. The environment is partially-known and updated in real-time to both the UAV and the FW. A pop-up threat may arise en route to the search site, which is currently unknown.

Mission Narrative:

1: The FW vehicle is commanded to gather information and possibly destroy an enemy site located in unknown territory. Because of the mission risk, the FW vehicle assigns a UAV, stored in a nearby airspace volume, to gather information at the designated site. The UAV leaves the loiter area and moves toward the designated task area. The F-15 follows behind at a higher altitude and safe distance.

2: The UAV is informed of a pop-up threat en route to the task area. The UAV accounts for the threat dynamically, automatically generates a revised safe trajectory around the threat and other no-fly zones, while notifying the FW vehicle of the threat's position.

3: As the UAV moves within a few minutes of the task location, the UAV notifies the FW vehicle of its location. At this point, the FW vehicle provides the UAV with the exact ingress and egress conditions into the site area. The UAV modifies its flight path to arrive at the site as commanded.

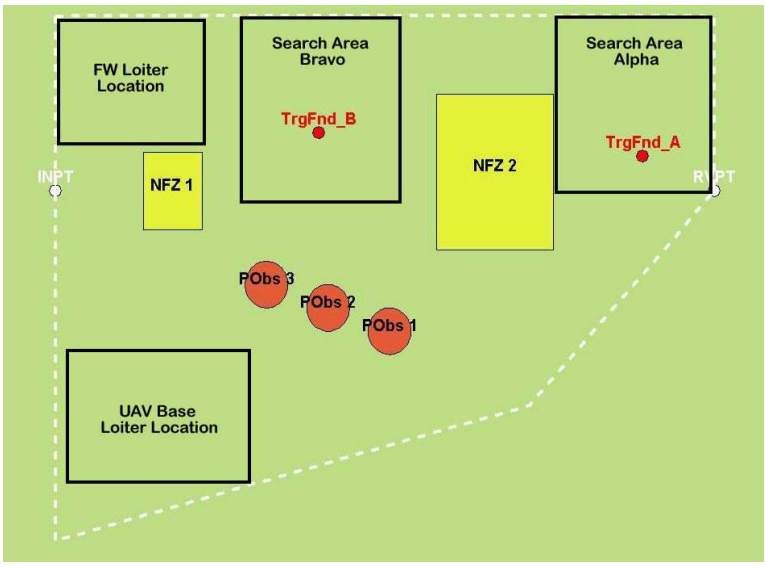


Figure 6. Sample Scenario Map for MIT SEC Capstone Demonstration

4: The UAV enters the site, notifies the FW of its location and begins its search for the target.

5: The UAV identifies the target and sends an image to the FW for evaluation. The FW commands the UAV to return to its original loiter area, while the FW prosecutes the target.

Exit Conditions: The UAV safely returns to the original pre-defined loiter pattern location, and the FW vehicle will return to flying an air-patrol near the enemy territory.

Using this narrative we designed a variety of sample scenarios, similar to the one depicted in Figure 6, to test our mission system. In this scenario there are two pre-determined No-Fly Zones (listed as "NFZ 1" and "NFZ 2") and three potential pop-up threats (denoted by "P Obs 1," "P Obs 2," and "P Obs 3"), which can be activated during the demonstration. In addition, there are two mission task areas (labeled "Search Area Alpha" and "Search Area Bravo"). Each task area has three potential ingress conditions which can be selected by the FW WSO before the vehicle reaches the task area location. Each task area also includes a threat/target (denoted by "TrgFnd A" and "TrgFnd B") which the UAV searches for and/or locates during the mission. Finally, the UAV starts the mission from the UAV Base Loiter location in the southwest corner of the flight area, and the FW vehicle maintains a loiter pattern near the northern border of the flight area area until the target has been detected, as shown in Figure 6. This scenario was developed for and used in the SEC Capstone Flight Test Demonstration, which took place during the last two weeks of June 2004.

A. FW Demonstration System

As mentioned before, the Fixed-Wing Demonstration System is designed to provide the F-15 Weapons System Officer (WSO) with the ability to communicate with the UAV using natural language commands. Based on the information above, we developed the block diagram in Figure 7 showing the high-level implementation of the Natural Language Interface system. In this figure, the Natural Language Interface module is confined to the FW Demonstration System. Since transmitting sentences over Link-16 requires a variable amount of bandwidth, we developed the system to interpret sentence commands into a nine data word message. This coding scheme meets user bandwidth constraints and simplifies the structure of the user-defined data sent between the UAV and FW vehicle during the demonstration.

Here, the Natural Language (NL) Interface module has two major independent components. The first component is designed to take sentence commands from the F-15 WSO and turn them into a coded command to be sent to the UAV over Link-16. The second component takes a coded command set from the UAV and

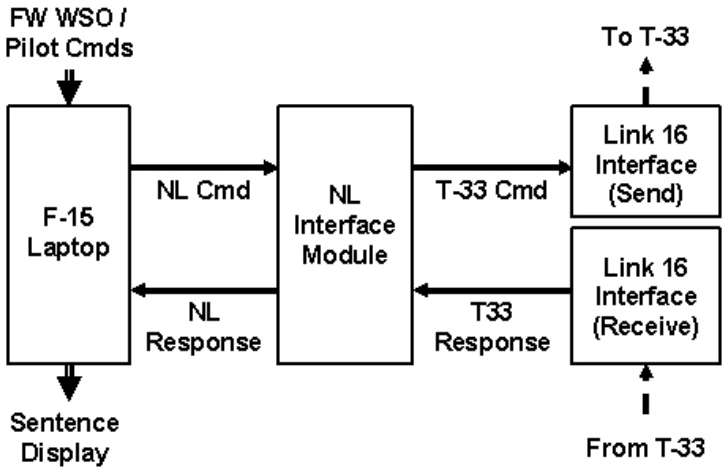


Figure 7. F-15 High-Level System Block Diagram

converts it into a natural language command response for the F-15 WSO to interpret. The core NL functions used in both components were developed together with the Teragram Corporation[12] based on demonstration requirements.

In order to implement the FW Demonstration system with the natural language parser, we first had to resolve a number of issues related to the real-time aspects of this system. For example, as mentioned before, we had to meet bandwidth constraints on the message sizes sent between both vehicles. In addition, the system had to reliably complete any actions in under one second to meet the real-time constraints on the system. Most importantly, both the Weapons System Officer and the UAV Mission System had to understand the context of the mission goals and communicate their respective actions to one another throughout the mission *in real-time*. Therefore, if a message was dropped on either side of the communications interface, both the operator and the vehicle had to be able to continue with their mission goals *regardless* of the communications link status.

From these requirements, we developed a script of the possible commands to be sent by the F-15 WSO during the flight experiment. These sentences can be selected by the F-15 WSO using a set of pre-defined experiment keys on the F-15 Laptop. When an experiment key is pressed, the associated sentence is sent to the Natural Language interface module. Each sentence is pre-processed, parsed and converted by the core NL functions into a nine number code to be used by the UAV as an input command. Messages sent between the vehicles follow the following protocol:

Message Description

Cmd ID: <*long integer*>

Cmd Data Words 1-8: <*double*>

First, each Command Identification (Cmd ID) value is used to denote different commands sent between the vehicle. For example, Cmd ID 102 may represent the "Command Acknowledge" data set, where as Cmd ID 106 may represent the "New / Change Current Task" data set. Second, because of user bandwidth limitations, we developed each command word to identify a maximum of eight data words for this demonstration. Next, when the UAV sends a command response, obstacle/threat warning or a command acknowledgement to the F-15, the NL Module is designed to save the command and send a conversation introduction to the F-15 WSO before the actual command is sent. Finally, the Natural Language interface architecture was designed to work directly with the Task Scheduling and Communications interface architecture on the UAV to ensure that commands from the NL interface are interpreted correctly and the appropriate response was provided. This design simplifies the communications protocol for the demonstration and reduces the amount of traffic over the communications link.

B. UAV Demonstration System

The UAV Demonstration System is designed to provide the following real-time capabilities:

- Safe autonomous vehicle navigation
- Dynamic task and mission planning and execution
- Reactive trajectory planning

Together with the Natural Language interface, this system provides flexibility for a mission operator to insert and change mission tasks during the operation. In addition, the Mixed-Integer Linear Programming-based guidance algorithms provide safe and reactive trajectory planning around obstacles and threats (both known and pop-up). Based on our mission requirements, we developed the block diagram in Figure 8 showing the high-level implementation of the MIT UAV Demonstration system.

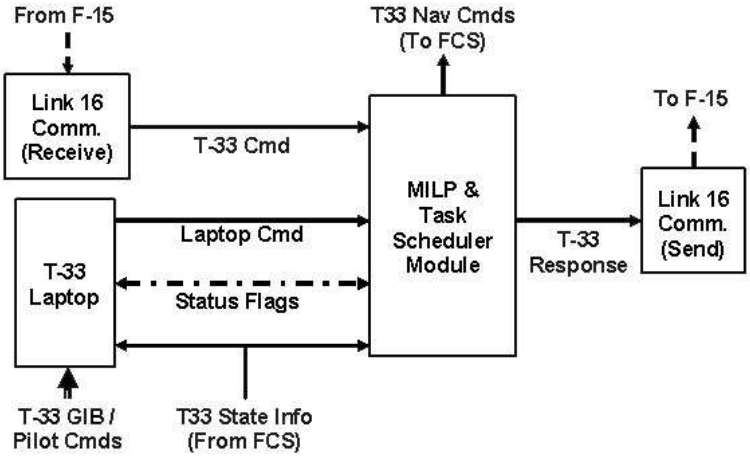


Figure 8. T-33 High-Level System Block Diagram

In Figure 8, notice that the Task Scheduling and MILP-based guidance modules have been implemented in the same code module for this demonstration, though each system component can be implemented on separate computers if necessary. The unified module (as it appears in Figure 8) is implemented in the following order:

- Input / Output Processing (includes Link-16 Comm. Processing)
- Pre-Processing and Task Scheduling
- Trajectory Generation (using MILP algorithms)
- Waypoint Post-Processing and Vehicle Command Calculations

In the unified module, the task scheduler became the pre-processing step for the MILP-based guidance algorithms. Using this approach, the navigation algorithms only receive information about the surrounding environment (i.e., obstacles, threats), the current state and position of the vehicle (altitude, velocity, position) and the next waypoint for the vehicle (as determined by the task scheduler). This arrangement reduces the amount of information the guidance algorithms must maintain in memory and separates the mission level tasks from the navigation tasks effectively.

Finally, the UAV mission system modules run nominally at 1 Hz. The trajectory planning algorithm step is nominally invoked every 10 seconds. If there is a change in the environment (i.e., a new task or a threat/obstacle), the MILP-based trajectory planning algorithm is invoked to provide an updated solution based on the new environment information.

VIII. System Development, Integration and Laboratory Testing

Beginning in late May 2003, we created a development and testing schedule to meet our system integration goals. A Simulation-In-the-Loop (SIL) laboratory test platform was built at MIT, as shown in Figure 9, to aid in the development of our demonstration software. This platform is composed of four core computers: two desktop computers execute the FW and UAV vehicle models and experiment control applications; two laptop computers (similar to the laptops used in the test flight experiments) manage the MIT FW and UAV SEC Demonstration Control Interface software. Later, this setup was expanded by adding two additional computers to separate the experiment control and vehicle simulation applications on different machines. Using wireless laptops connected to our laboratory LAN, we have been able to simulate command latency and other real-time issues during the development and verification testing phases of this project.

Figure 9. MIT SEC Demonstration Simulation-In-the-Loop (SIL) Laboratory Setup

Several intermediate tests were used to evaluate each module under demonstration-like conditions. Test conditions included communications link latency, message drop-outs, invalid experiment key selection, data scaling issues and modeling errors. In addition, we tested the trajectory planning algorithms in real-time with a Matlab Simulink test program.

From late November 2003 to mid-March 2004, the integration phase of the project began. Using our SIL laboratory, we tested our FW and UAV mission software with the Boeing DemoSim vehicle simulation application in real-time. Figure 10 shows one of the many integrated demonstration initialization tests. As the FW and UAV vehicles approach the flight area, we wanted to test all aspects of our demonstration software before starting the experiment. Therefore, after the MIT UAV demonstration software is initialized and the T-33 operators turn vehicle control over the MIT UAV Demonstration software, the vehicle will automatically fly to the UAV Base Loiter Location and command the vehicle to the initial conditions for the MIT flight experiment. This provides the T-33 operator crew with flexibility in initializing the MIT mission software during the demonstration. In this figure, notice that the UAV will also send status messages to the FW WSO to initialize the MIT FW Demonstration software, thus verifying the integrity of the Link-16 Communications interface before starting the demonstration mission.

Next, Figure 11 shows one of the pop-up obstacle avoidance tests used to verify the integrity of the integrated MIT SEC Demonstration mission software using the "Safe Trajectory" algorithm. In this test two pop-up obstacles were placed into the demonstration area as the UAV was en route to Search Area Alpha. This test highlighted the Safe Trajectory algorithm's ability to develop safe, dynamically feasible paths for the vehicle after unexpected changes to the environment. When the pop-up obstacle/threat is inserted into the demonstration area, the UAV automatically sends a status message to the FW WSO confirming that it observed a new threat in the environment. In addition, the UAV inquires if it should continue on the current mission. Next, the FW WSO can decide to send the UAV back to its base loiter location or command the vehicle to proceed with its current mission task. Notice in Figure 11 that as the second pop-up obstacle is inserted in the environment when the UAV is south of the first pop-up obstacle (designated "P Obs 3" in

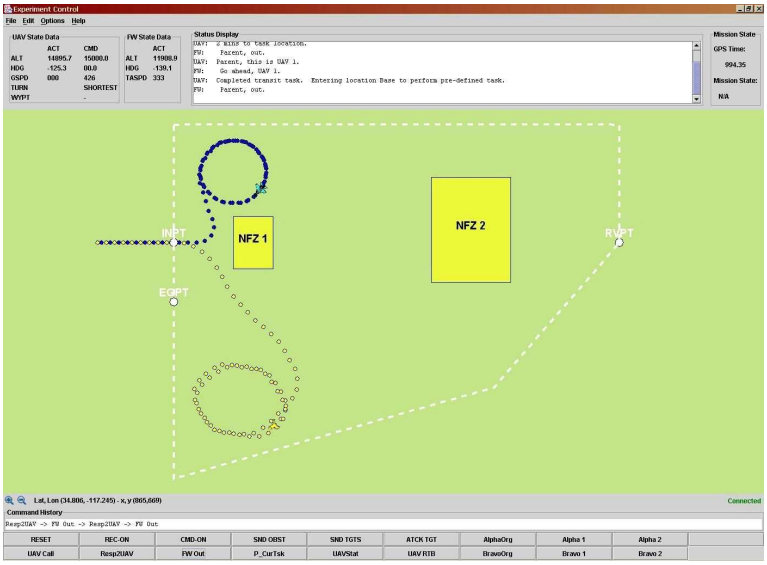


Figure 10. MIT SIL Test 1 - Initialize Demonstration

the figure), the vehicle changes its original trajectory to immediately turn left and proceed northeast over the second pop-up obstacle (designated "P Obs 1" in the figure). After passing the pop-up obstacles, the vehicle levels out and flies at a safe distance from No-Fly Zone 2 (NFZ 2) before turning north to enter Task Area Alpha to perform a search task.

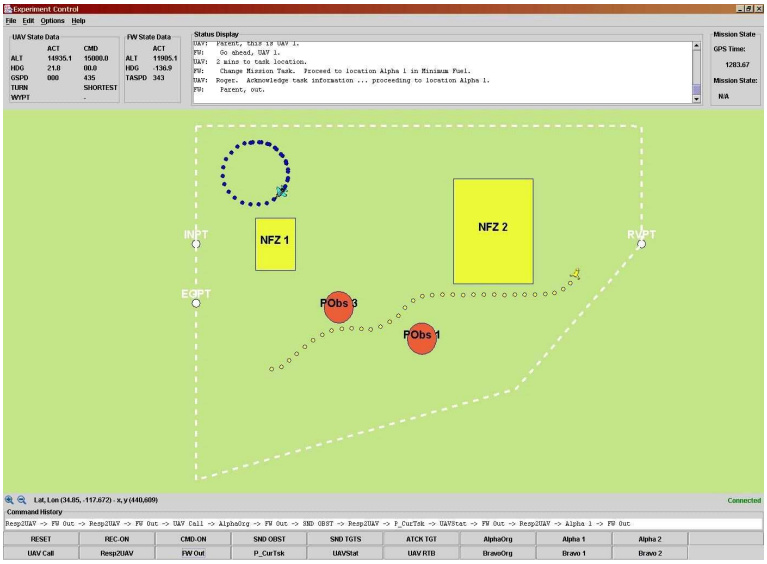


Figure 11. MIT SIL Test 2 - Safely Avoid Two Pop-up Obstacles

From mid-March 2004 to mid-April 2004, we performed a series of verification and validation tests on the final MIT demonstration mission software using the MIT SIL laboratory setup. We logged over 300 hours of testing on our final software version to detect software anomalies, test our flight demonstration mission plan and validate the integrity of the integrated system for a number of conditions. For example, since this mission system was developed so that multiple experiments could be performed by the system without resetting the experiment control software, we developed a series of tests to validate the performance of the system when multiple mission tasks are issued in sequence.

Figure 12 shows a test where the UAV was commanded to fly two consecutive missions from the UAV

Base Loiter Location using the "Safe Trajectory" algorithm. This figure shows the UAV's coverage over both search areas during the search task portions of the mission. The UAV safely avoided two pop-up obstacles en route to each search area location. The main objective of this test was to ensure that the UAV will return to the base loiter location after it finishes the search task (provided that another task was not given to the UAV). First, the FW WSO commands the UAV to proceed to one of the search areas, but does not issue the Return-To-Base (RTB) loiter command to the UAV during the mission. As shown in Figure 12, after the UAV completes its search pattern in either of the task areas, it proceeds to the next scheduled task – which in each case is RTB – if no additional tasks were scheduled. After finishing its search of the task area, the UAV informs the FW WSO that it has completed the search task and will proceed back to the Base Loiter location and await another set of commands. This software provides the vehicle operators, pilots and test directors with flexibility in task and mission management during the MIT flight experiment.

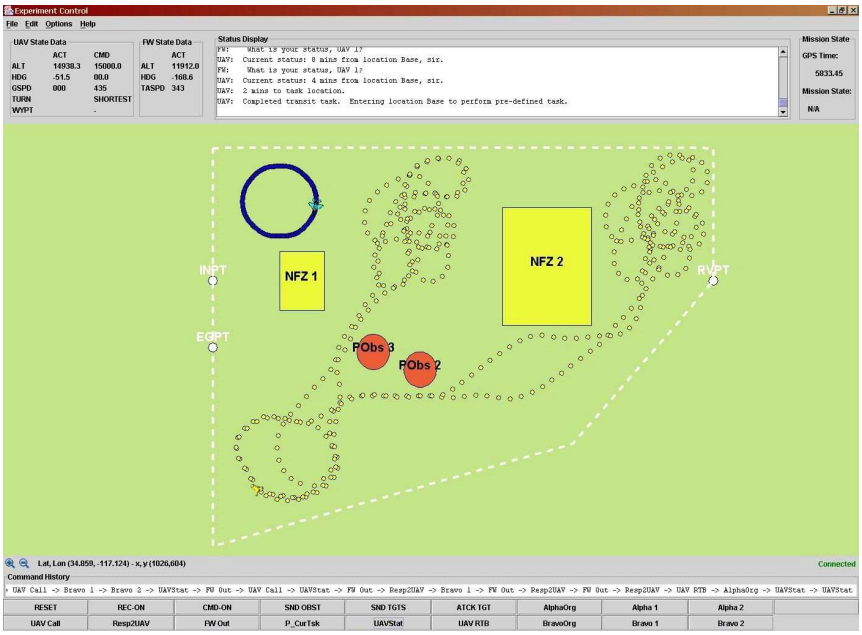


Figure 12. MIT SIL Test 3 - Two Consecutive Missions without Reset

IX. Flight Test Results

From mid-April 2004 to mid-June 2004, the final demonstration software build was turned over to Boeing Phantom Works in St. Louis for Verification and Validation testing on the HIL simulator. After successfully completing this testing in the lab, the software was successfully transitioned to the actual vehicle hardware test setup for testing at NASA Dryden in late June 2004. During this test period multiple successful F-15/T-33 sorties were flown and the MIT Mission Software package was tested three times using the MILP "Safe Trajectory" algorithm. The MIT SEC Capstone Demonstration software performed well in each of these experiments. These flight experiment marks the first time that a Natural Language Interface has been used by a ground operator and a manned vehicle to task and command a UAV in real-time. This flight test also marks the first time that a Mixed-Integer Linear Programming-based Guidance System has been used to control a UAV in coordination with a manned vehicle. Overall, each of the three missions verified the capabilities of the MIT Mission software as witnessed in the laboratory and demonstrated the flexibility of the MIT Mission software allowed the test team to make adjustments in real-time. The actual results from the flight test will be provided in a future paper.

X. Conclusion

In this paper we have described and demonstrated our design, integration and testing approach for a manned vehicle-UAV mission system in a partially-known environment in real-time. From our tests, we have demonstrated that combining the task scheduling algorithm with the Natural Language interpreter can be used to enable a Weapons Systems Officer on-board a fighter aircraft to successfully command a UAV during a mission in real-time. In addition, Mixed-Integer Linear Programming-based trajectory planning algorithms have been successfully tested and executed in a real-time simulation environment. After a successful testing effort in the MIT SEC SIL laboratory setup, the software was transitioned to Boeing and successfully tested in their HIL simulator before flight. During flight testing, multiple successful F-15/T-33 sorties were flown and the MIT mission software was tested successfully on three separate flights.

Currently, we are beginning to transition this work to other multiple vehicle platforms. The results from this research and flight testing have been very helpful in the design of multiple vehicle platforms in sustained operational environments. We hope to use the results from this project to help investigate issues dealing with safe trajectory planning and integrated natural language-based communication systems for multiple autonomous vehicle environments. We believe that in the future Natural Language interfaces will be the most efficient way to communicate with unmanned vehicle systems. Finally, this research can be used to aid in the development of real-time operational management systems for multiple vehicles in dynamic and partially-known environments.

Acknowledgments

This research was funded by DARPA (Software Enabled Control) F33615-01-C-1850.

References

[1]Wohletz, J., Castanon, D., and Curry, M., "Closed-Loop Control for Joint Air Operations," *Proceedings from the American Control Conference*, Arlington, VA, June 2001.

[2]Paunicka, J. L., Mendel, B. R., and Corman, D. E., "The OCP - An Open Middleware Solution for Embedded Systems," *Proceedings of the 2001 American Control Conference*, Arlington, VA, June 2001.

[3]Floudas, C. A., *Nonlinear and Mixed-Integer Programming - Fundamentals and Applications*, Oxford University Press, 1995.

[4]Schouwenaars, T., Moor, B. D., Feron, E., and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," *Proc. of the 2001 European Control Conference*, Porto, Portugal, September 2001.

[5]Schouwenaars, T., How, J., and Feron, E., "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proceedings of the 2004 American Control Conference*, Boston, MA, July 2004.

[6]Bellingham, J., *Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming*, Master's thesis, Massachusetts Institute of Technology, 2002.

[7]Richards, A., Kuwata, Y., and How, J., "Experimental Demonstrations of Real-time MILP Control," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Austin, TX, August 2003.

[8]Bellingham, J., Richards, A., and How, J., "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the IEEE American Control Conference*, May 2002.

[9]Kuwata, Y. and How, J., "Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control," *Proceedings of the 2004 American Control Conference*, Boston, MA, July 2004.

[10]ILOG, Mountain View, CA, *ILOG Optimization Suite: White Paper*, 2001.

[11]Mettler, B., Valenti, M., Schouwenaars, T., Kuwata, Y., How, J., Paunicka, J., and Feron, E., "Autonomous UAV Guidance Build-up: Flight Test Demonstration and Evaluation Plan," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, TX, August 2003.

[12]Teragram Corporation, 10 Fawcett Street, Cambridge, MA 02138, *http://www.teragram.com*.