Human-robot interaction for field operation of an autonomous helicopter

Henry L. Jones [a], Eric W. Frew [b], Bruce R. Woodley [c], Stephen M. Rock [d]

Aerospace Robotics Laboratory [e], Stanford University
Durand Building, Room 250, Stanford, C 94305

ABSTRACT

The robustness of autonomous robotic systems to unanticipated circumstances is typically insufficient for use in the field. The many skills of a human user often fill this gap in robotic capability. To incorporate the human into the system, a useful interaction between man and machine must exist. This interaction should enable useful communication to be exchanged in a natural way between human and robot on a variety of levels.

This paper describes the current human-robot interaction for the Stanford HUMMINGBIRD autonomous helicopter. In particular, the paper discusses the elements of the system that enable multiple levels of communication. An intelligent system agent manages the different inputs given to the helicopter. An advanced user interface gives the user and helicopter a method for exchanging useful information. Using this human-robot interaction, the HUMMINGBIRD has carried out various autonomous search, tracking, and retrieval missions.

Keywords: Helicopter, human, interaction, interface, robot, supervisory control,telerobotics, usabilit

1. INTRODUCTION

Research using field robots has begun to gain momentum in recent years due to a number of emerging technologies such a the Global Positioning System (GPS), small computers, and machine vision. These robots that operate in an unstructured environment promise to allow real missions to be performed with much greater ease and safety than is currently possible. One particular type of field robot, the autonomous unmanned aerial vehicle (UAV), has already been used in significant numbers. DarkStar, Predator, and the Unmanned Tactical Aircraft (UTA) are current military UAV programs. Most civilian uses of UAVs fall into one or more of three specific categories – dirty, dull, or dangerous. These potential application include remote surveying and aerial mapping, power line inspection, crop dusting, fire fighting, movie filming, and surveillance.

One subject that is still rather new is that of the interaction between field robots and their users. Preliminary work in the UTA program found that the human/system interface was the most difficult technical issue. [1] This does not imply that interfaces for robots have not been researched; indeed, many new display and interface technologies have been created or adapted for the control of robots.[2] However, bringing these new technologies to working field robots continues to be a challenge.

The related field of human-computer interaction (HCI) has been steadily growing in recent years, and has followed hardware and then software as the focus of computer development.[3] However, even the definition of human-robot interaction is mostl unsettled.[4] Many of the lessons learned in HCI should be useful in human-robot interaction. Of course, there are particula differences between computers and their physical cousins. Foremost among them is the ability of robots to move beyond the electronic domain and into the physical world.[5] Nonetheless, an important discovery of HCI researchers was that the fastest method of technical advancement was to implement new interactions and evaluate their performance.[6]

Since 1995 the Stanford Aerospace Robotics Laboratory (ARL) has been operating a small helicopter, HUMMINGBIRD, capable of fully autonomous operation as part of a program of basic research to develop capabilities for UAVs. Initial work in this program focused on demonstrating the feasibility of using Carrier-Phase Differential GPS (CDGPS) as a sensor system for attitude and position control as well as navigation. Precision flight was experimentally demonstrated b performing autonomous hover, autonomous retrieval of a ferromagnetic disk using a magnet-tether manipulator, and autonomous landing tasks.[7]

[a] Ph.D. Candidate, Department of Aeronautics and Astronautics
[b] Ph.D. Candidate, Department of Aeronautics and Astronautics
[c] Ph.D. Candidate, Department of Electrical Engineering
[d] Professor, Department of Aeronautics and Astronautics
[e] (Correspondence): Email: heli@arl.stanford.edu; WWW: http://arl.stanford.edu/~heli; Telephone: (650) 723-3608; Fax: (650) 725-3377

The concentration on autonomy was a result of the requirements of the Aerial Robotics Competition (ARC) of the Association for Unmanned Vehicle Systems International. ARC rules prohibited any communication from a human, and consequently the HUMMINGBIRD system employed no in-flight interaction.[8] A list of instructions was given to the helicopter at startup and parsed in the air. A highly structured environment made such direct *a priori* instruction practical.

The addition of computer vision processing to HUMMINGBIRD allowed tasks which required the helicopter to sense th objects of interest in the environment whose *a priori* GPS locations were unknown. For example, a search task was performed in which the objective was to find and then track an object.[9] A live-video picture was sent to a ground station, as well as basic data. The user was able to see video from the point of view of the helicopter and to monitor its GPS sensors and task state. However, no communication was possible in the ground-to-air direction. The helicopter continued to parse an instruction list given at startup and operate autonomously.

Multiple tests revealed some of the constraints posed b *a priori* instruction and autonomous operation. At times, HUMMINGBIRD completed the given instructions flawlessly yet did not satisfactorily accomplish the general mission objectives. In other cases the mission was completed ahead of schedule and additional tasks could have been accomplished. Ground operators were frustrated by a lack of input on any of the helicopter's operation levels. As the basic functionality of the helicopter had been advanced to a stage that allowed meaningful tasks and missions to be performed, th HUMMINGBIRD project began work to make the entire system more useful. A two-way interface on the ground was created and the onboard system was changed to allow dynamic task direction and execution. Described below are the results of the current research activities in the Stanford ARL that have addressed these issues of allowing user interaction with a small semi-autonomous helicopter.

2. DESCRIPTION OF PHYSICAL AND FUNCTIONAL SYSTEMS

2.1 Physical system

2.1.1 Helicopter

The HUMMINGBIRD helicopter is a heavily modified Schluter Futura model helicopter (Figure 1). It was selected for its low empty weight, large payload capacity (over 25 pounds), and fifteen minute endurance. The modifications included stiffening of the structure to accommodate the mounting of the video and electronic components and the replacement of its standard two-horsepower single cylinder engine by a five-horsepower dual-cylinder engine intended for use on model airplanes. This required the design and construction of a new engine mounting plate and modification of the power transmission devices.

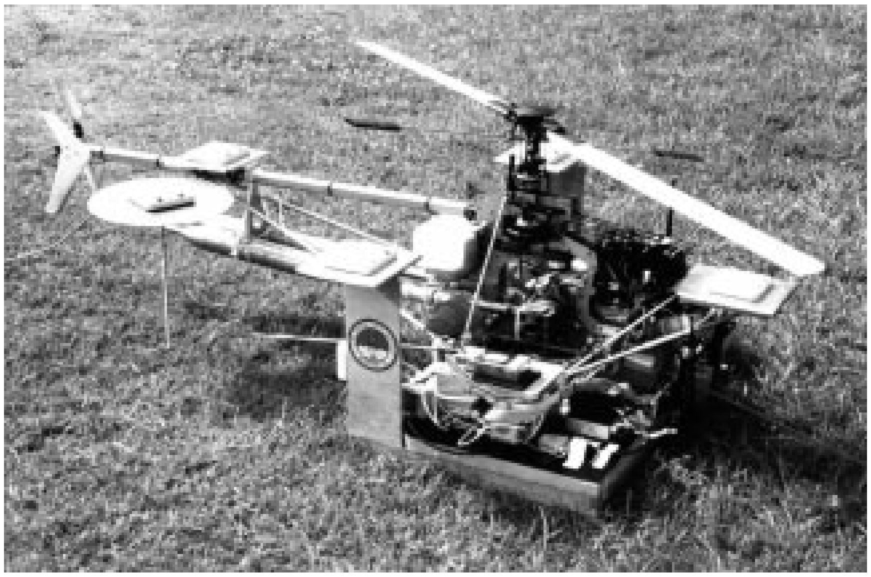


Figure 1. The HUMMINGBIRD helicopte

The Futura is equipped with two stabilization devices -- Hiller paddles and a mechanical rate gyro. The Hiller paddles are used to slow the lateral and longitudinal dynamics. The rate gyro is used in a simple electrical feedback loop to slow the yaw dynamics. Both of these devices are standard equipment for helicopters of this size, and are essential to permit manual flight operation.

The computing and sensing components aboard the HUMMINGBIRD vehicle include two GPS receivers, four GPS antennae, two video cameras, a wireless video link, a wireless Ethernet link, a 9600-baud communications link, a 486-class computer and two HC11 computers. The supporting ground station includes a GPS antenna and receiver and a dual-Pentium computer for vision processing, data recording, and the user interface.

2.1.2 Global Positioning System

The HUMMINGBIRD helicopter uses the Global Positioning System (GPS) as its primary sensor, both for navigation and stabilization. This is a satellite-based navigation system that offers absolute positioning in a world-fixed reference frame anywhere on the globe. The Carrier-Phase Differential GPS (CDGPS) technique employed by the ARL computes the attitude and position of the helicopter at 10 Hz, which is fast enough for the stabilization of the helicopter dynamics. GPS offers many additional advantages for an airborne navigation system, including (1) integration of all sensors into a single unit; (2) drift-free rate information; (3) no moving parts; and (4) relatively small size and power consumption. Previous ARL papers discuss the details of the use of CDGPS in control.[9, 10]

2.1.3 Vision sensing

The HUMMINGBIRD vision system consists of a pair of downward pointing Sony XC-999 color cameras mounted on the front of the helicopter. Due to the weight constraints of the vehicle, the vision processing is done on an off-board ground station computer. Two wireless video transmitter units are used to send the color images from the helicopter to a dual Pentium computer. After processing, the information is telemetered to the onboard computer via a wireless Ethernet link.

2.2 Functional system

2.2.1 Control

The control system for the HUMMINGBIRD helicopter is comprised of two loops. A high-bandwidth inner loop provides attitude stabilization and vehicle position control based entirely on the CDGPS system. A low-bandwidth outer loop provides position, velocity, and acceleration commands to the inner loop based on the current task being executed. The development of this outer loop will be discussed in a Section 3.2.

The inner loop block diagram is shown in Figure 2. Low-level control of the helicopter is carried out using an LQR/LQE (LQG) controller that operates at 64 Hz. Although full-state feedback is possible using only the CDGPS system, the LQE i used to smooth the output of the GPS sensor. Because the sensed data (position, attitude, and engine rpm) arrives with varying time delay, each piece of information includes the time at which it was created. All time delays are explicitl accommodated by propagating the helicopter states forward in time with a dynamic vehicle model.

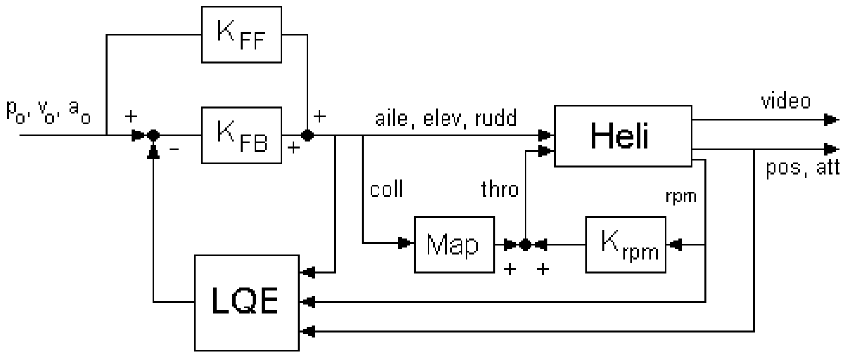


Figure 2. Inner Loop

2.2.2 Vision processing

The vision processing is performed by a Teleos Corporation (now a division of Autodesk) Advanced Vision Platform (AVP) system. The HUMMINGBIRD uses the YUV color segmentation ability of AVP to identify objects. The segmented UV images correspond closely to red and blue color components and test objects of these colors are used in order to simplify processing. This color-based object identification is performed on both the left and right camera images. The use of objects of unique color solves the stereo correspondence problem. With the coordinates of the object in both images, stereo triangulation is used to calculate the location of the object relative to the fixed camera reference frame. This information is then sent through the Ethernet link to the onboard flight computer, which transforms the location into the proper helicopte reference frame. This system has an accuracy of approximately 5 cm in range and 2 cm in location at a range of 2 meters. The object-tracking algorithm runs at 10 Hz.

3. USER INTERACTION

3.1 Object-based task-level control

The Stanford Aerospace Robotics Laboratory has developed and implemented Object-Based Task-Level Control (OBTLC), a robotic control architecture that is inherently user-centered. Originally developed for use with remote robotic systems fo space applications,[11] OBTLC has been employed in space robot simulators,[12] flexible manipulators,[13] factory workcells [14] and underwater vehicles.[15] The philosophy of OBTLC has three main tenets:

- modern technology cannot produce a substitute for human intelligence,
- automatic feedback control is unaffected by any limits in human bandwidth, endurance, or capacity to manage multiple tasks simultaneously, and
- when humans and computers are insightfully integrated, the resulting robotic system is more effective than if either were used alone [15]

The phrase “object-based task-level control” describes the interaction between the human and the robot. “Object-based” designates the focus of the human supervisor’s attention when using the robot. Rather than concentrating on *how* to control the robot by driving the low-level actuation, the operator perceives the situation at hand and formulates *what* needs to happen to accomplish the system-level goals. The term “task-level” signifies the level of human interaction. The information passed from the human to the machine is a command that may range from “move left” to “follow the red object.” The robot then follows this instruction autonomously. As a result of this division of labor, OBTLC has been shown to be an attractive alternative to the use of teleoperation or strict supervisory control.[12, 16]

A key component of OBTLC implementation is the human/robot team. Ideally, humans and machines would coexist in a unified system with the duties of each member optimized with respect to their inherent abilities. Human intelligence is more capable of inferring information from scarce telemetry, setting goals and planning missions, and responding appropriately to unexpected situations. At the same time, computers are capable of the many chores necessary to control robots at a low level, such as controlling multi-input/multi-output systems or processing large amounts of sensed data. However, this ideal relationship is very difficult to realize.[17]

The HUMMINGBIRD helicopter may be directed in four distinct ways: an *a priori* instruction list, in-flight list manipulation, position control via a joystick, and direct teleoperation. These modes give the human a variety of options for accomplishing a mission and they enable the full range of input that makes OBTLC possible. In any system, this variety of control levels should be advantageous and possible without excessive additional overhead.[2] The normal operation mode is full autonomy in which the list of instructions is carried out sequentially by the helicopter. Task-level direction is possible throughout the flight by editing this task list from the ground. Lower-level position-only commands are possible through the joystick teleoperation task. Although a more continuous range of direction would be more consistent with the theory of OBTLC, the HUMMINGBIRD will enable sufficient investigation into the human/robot team component of OBTLC when implemented on a field robot. The development of the system to enable such operation is explained below.

3.2 System agent

The low-bandwidth outer loop shown in Figure 3 creates kinematic position, velocity, and acceleration reference command for the inner loop to track. The formation of these set points depends on the current action of the helicopter. Because the many different actions require a variety of inputs and operate independently, an autonomous System Agent was created to manage the diverse input and output. At each cycle through the high-bandwidth inner loop, the controller asks the Agent fo the correct current set points. In this manner, complex sequences of actions can proceed seamlessly.

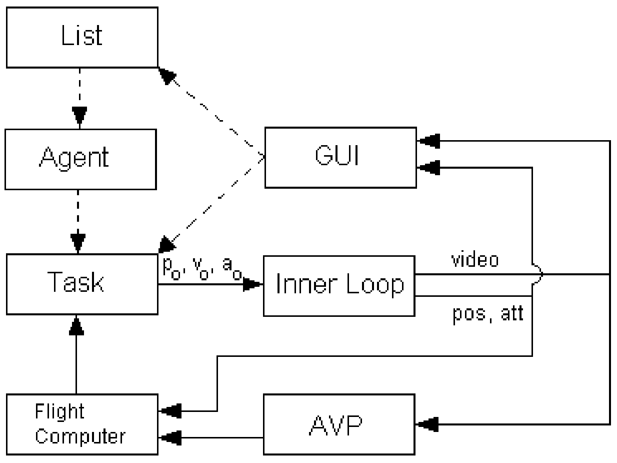


Figure 3. Outer Loop

The System Agent contains an instance of a C++ virtual base class named **Task** (Figure 4). The only member function of **Tas** is the virtual function *Execute()*. Through the Agent's instance of **Task**, the inner loop receives its commands – a call of *Execute()* by the inner loop at each timestep sets the proper values for the desired position, velocity, and acceleration. To accommodate a variety of actions, instances of derived classes of **Task** redefine *Execute()*. The nature of this construction is shown in Figure 4. The system agent uses instances of these derived classes to provide substance to the *Execute()* call.

For example, if a **Follow** task is active, the *Task.Execute()* call will be carried out b **Follow**'s own *Execute()* function, and may call other internal functions such as *FindObject()*. The System Agent makes sure that a valid instance of one of these derived classes is running at all times. This organization allows project developers to create and incorporate new and more complicated tasks easily. The base class provides the necessary framework, while each derived class implements *Execute()* according to its particular purpose. Future work will develop greater depth and breadth for this tree.

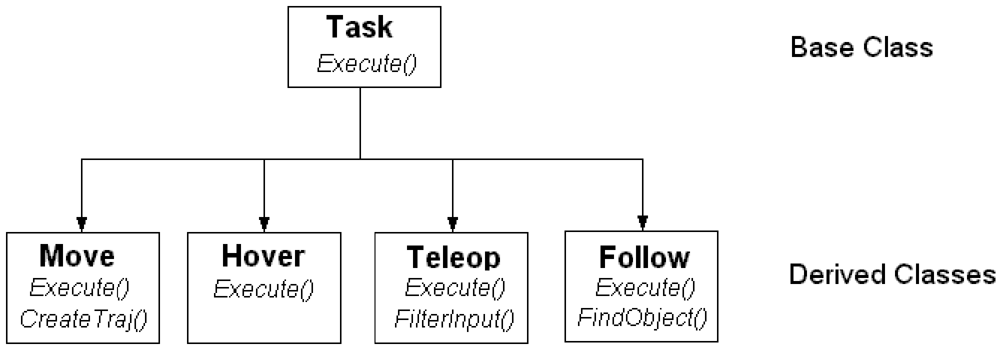


Figure 4. Base and derived task classes

During flight, the Agent follows the list of instructions maintained on the helicopter. When autonomous operation begins, the Agent executes the first instruction on the list. The name of the instruction tells the agent which task to begin, and the associated data provide the distance, velocity, and/or length of time to use during execution. The Agent constructs an instance of the appropriate derived class, as discussed above, with the data used as arguments for the constructor. When the task ends, this process repeats. If no further instructions exist, the Agent creates a **Hover** task at the current position of the helicopter.

The different types of tasks require a variety of forms of data exchange with the rest of the system. For example, a simple motion task between GPS locations only needs to be given the necessary change in position. On the other hand, an autonomous tracking task requires continuous knowledge of the current locations of the helicopter and target. Direct teleoperation (manual control) does not permit any computer control, yet joystick teleoperation requires automatic attitude control while giving the user control of the helicopter velocity. It would be virtually impossible for the individual tasks to negotiate among themselves to decide which input was required from the user or instruction list and what the output to the inner loop controller would be. The System Agent, therefore, serves a very important function by allowing the possibility of many control levels while presenting a consistent interface to the inner loop controller and the ground operator.

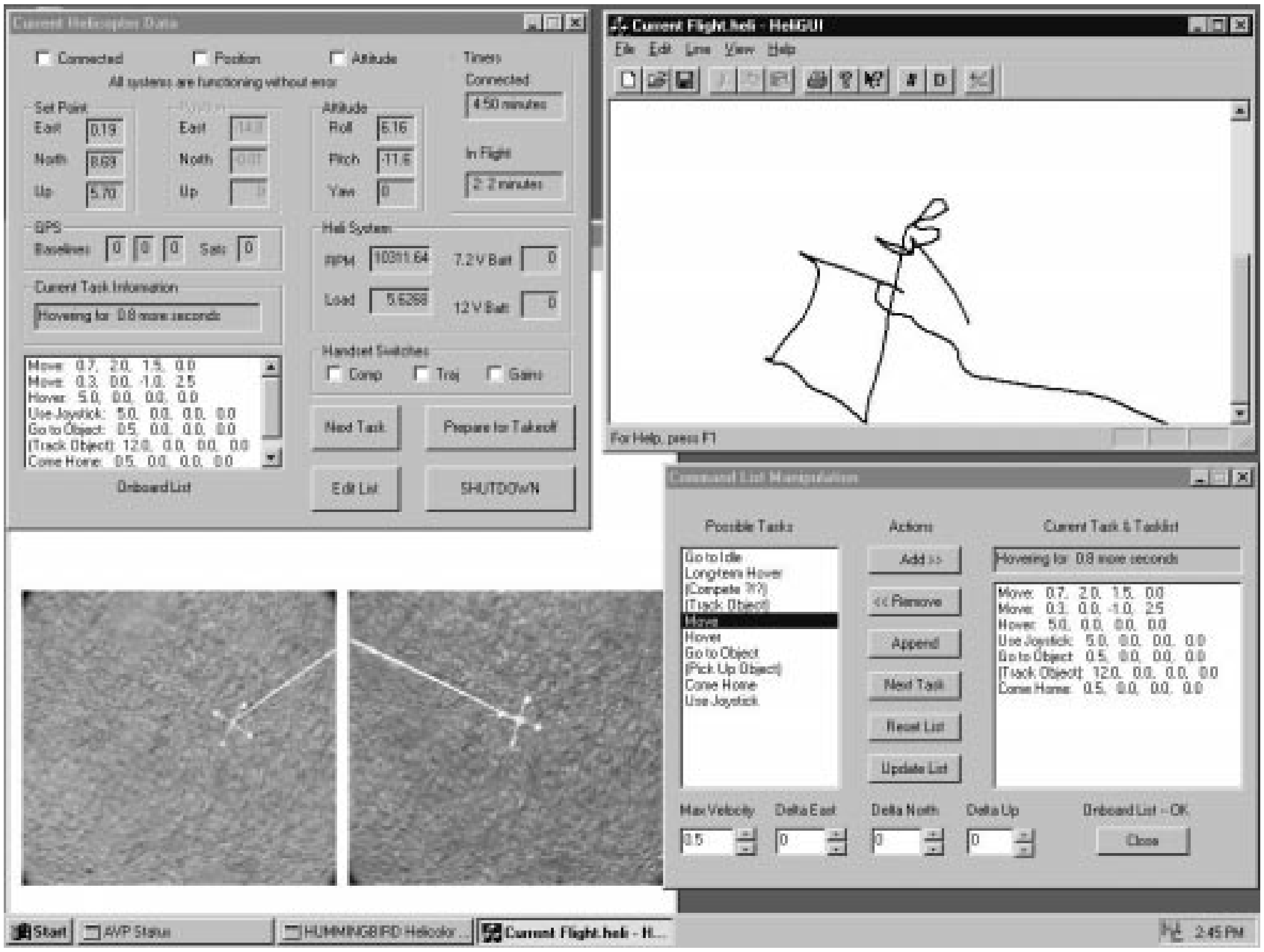
3.3 User interface

3.3.1 Preliminary interfaces

For the first HUMMINGBIRD prototype, the absence of an in-flight data link limited the indication of system state to only a blinking light on the helicopter tail boom. This was sufficient for simple autonomous operation. The next user interface, fo the current HUMMINGBIRD prototype, used a text-only display. A ground station operator monitored the helicopter by watching for any abnormalities in the displayed numbers. This required the undivided attention of the user. Two commands could be sent to the helicopter – one would signal that the flight was about to begin, and the other would signal that the flight had ended. For these preliminary interfaces, user interaction was extremely limited.

3.3.2 Current interface

The current interface for the HUMMINGBIRD, shown in Figure 5, was designed to allow graphic output, task-level and joystick input, and an easily readable appearance. Due to a number of factors, including the available development platform, operating system, software legacy, and programmer skills, Microsoft's Visual C++ 5.0 and the Microsoft Foundation Class (MFC) libraries were chosen as the development system for the new interface. Visual C++ and MFC allow the developer to rapidly produce the dialog boxes, menus, and toolbars that are familiar objects to window system users.



Current Helicopter Data

Connected Position Attitude

All systems are functioning without error

Timers
Connected: 4.50 minutes
In Flight: 2.2 minutes

Set Point
East: 0.19
North: 8.69
Up: 5.70

Attitude
Roll: 6.16
Pitch: -11.6
Yaw: 0

GPS
Baselines: 0 0 0 Sats: 0

Heli System
RPM: 10311.64
Load: 5.6288
7.2 V Batt: 0
12 V Batt: 0

Current Task Information: Hovering for 0.8 more seconds

Handset Switches: Comp Traj Gains

```
Move:  0.7,  2.0,  1.5,  0.0
Move:  0.3,  0.0, -1.0,  2.5
Hover:  5.0,  0.0,  0.0,  0.0
Use Joystick:  5.0,  0.0,  0.0,  0.0
Go to Object:  0.5,  0.0,  0.0,  0.0
(Track Object):  12.0,  0.0,  0.0,  0.0
Come Home:  0.5,  0.0,  0.0,  0.0
```

Onboard List

Next Task | Prepare for Takeoff | Edit List | SHUTDOWN

Possible Tasks:
Go to Idle
Longterm Hover
(Compete ?!?)
(Track Object)
Move
Hover
Go to Object
(Pick Up Object)
Come Home
Use Joystick

Actions: Add >> | << Remove | Append | Next Task | Reset List | Update List

Current Task & Tasklist: Hovering for 0.8 more seconds

```
Move:  0.7,  2.0,  1.5,  0.0
Move:  0.3,  0.0, -1.0,  2.5
Hover:  5.0,  0.0,  0.0,  0.0
Use Joystick:  5.0,  0.0,  0.0,  0.0
Go to Object:  0.5,  0.0,  0.0,  0.0
(Track Object):  12.0,  0.0,  0.0,  0.0
Come Home:  0.5,  0.0,  0.0,  0.0
```

Max Velocity: 0.5
Delta East: 0
Delta North: 0
Delta Up: 0
Onboard List -- OK
Close

Figure 5. Current HUMMINGBIRD Interface

The top left window in Figure 5 gives the user raw numerical information about the helicopter. In many ways this display is identical to the previous text-only interface. Only the most current data is shown, no input is allowed into this window, and time histories of the data are not available during flight. Previous values of importance must be seen and remembered by the user.

In the top right corner of Figure 5, an x-y plot of the actual helicopter path is continuously updated. This plot is particularly useful for monitoring the actual path of the helicopter during a search pattern to identify any potential blind spots. No other output to this plot is currently implemented, although efforts to display the positions or paths of objects sensed by the vision system are underway. This is the only output of historical data on the current interface.

A dialog box dedicated to manipulation of the instruction list onboard the helicopter is located in the bottom right corner of Figure 5. In the left box is a list of all possible instructions ("Possible Tasks") to send to the helicopter. On the right is the current helicopter list ("Current Tasklist"). The helicopter maintains the Current Tasklist, and a copy is sent down to the ground station whenever a change takes place on the helicopter. A formatted description of the task currently controlling HUMMINGBIRD is above the Current Tasklist. Consequently, the user sees the first instruction on the list "move up" to become the current task as the instruction is removed. Along the bottom of this dialog box are edit boxes that allow the user to specify the data that accompanies each task. The titles above each are sensitive to the current selection in the Possible List box.

The center of the task manipulation dialog box contains the different possible actions that the user can take on the instruction list. The buttons are intended to be self-explanatory. **Add** takes the user selection in the Possible List on the left and adds this instruction with the current data in the edit boxes at the position of the Current List selection. **Append** adds this information only to the end of the list. **Remove** deletes the selection from the Current List. **Next Task** prompts the Agent to process the first instruction on the list. **Reset List** reloads the original list so that the flight can be restarted. **Update List** makes sure that the onboard and ground lists are synchronized. In all cases, the appropriate command is sent directly to the helicopter and the copy of the instruction list on the ground is deleted. There is no output to the Current List box until the helicopter sends down a copy of the new list. In this way the user can be sure that the list being viewed corresponds exactl to the onboard list.

Finally, video images from each of the onboard cameras are seen in the bottom left corner of the screen in Figure 5. On each image, when applicable, is a crosshair that locates any objects of interest in the camera's view. This identification, a mentioned above, is based on the objects' color, and the user may change the color thresholds for the vision processing. To increase the processing speed, the image display rate has been slowed to one Hertz. Consequently, a separate television set (not shown) with a direct feed split from one camera is used adjacent to the computer screen. This allows the operator to use the video directly and not pay a penalty for the slow computing speed of the ground station. Faster computers would eliminate this need for a separate display in the future. In Figure 5, the magnetic retrieval device can be seen in each camera view.

4. CONCLUSION

The current HUMMINGBIRD human-robot interaction is much more bi-directional than before. Both parties are passing useful information to the other throughout the flight. The human can use her superior perception skills, knowledge of external information and constraints, and understanding of the entire system capabilities to guide the work in progress. The autonomous system is able to perform acts that would not be possible in person or through direct teleoperation. However, compared to human-human interactions, the HUMMINGBIRD system does not yet fit the typical definition of a team as required by OBTLC. Rather than a pair of equals, the current human-robot interaction is better defined as an autocracy. Recent work in human-computer interaction has attempted to address this issue.[3] Further research with the HUMMINGBIRD is planned in this area.

The HUMMINGBIRD helicopter has been a continuously evolving platform that has steadily increased its capabilities. With each increase in functionality, greater pressure has been exerted on the user interaction to make the system equally usable. Experimental missions, especially with individuals of varying background and capabilities, utilizing the new interface and agent should yield adequate guidance for further work. Theoretical bases for the design decisions have been difficult to find. Consequently, this project will continue to use its unique equipment to establish guidelines for future interaction design.

5. ACKNOWLEDGEMENTS

This work was partially funded by NASA Ames Research Center. The authors are grateful to NASA for their continued support of this research.

6. REFERENCES

1. Michael Francis, “The UTA: Redefining the air combat system,” *Aerospace America*, pp. 44-50, May 1998.
2. Thomas B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*, The MIT Press, Cambridge, 1992.
3. Jonathan Grudin, “The computer reaches out: The historical continuity of interface design,” *CHI ’90: Human factors in computing systems*, J.C. Chew & J. Whiteside (eds.), pp. 261-268, ACM Press, Seattle, 1990.
4. Yuichiro Anzai, “Human-robot-computer interaction: a new paradigm of research in robotics,”*Proceedings of the IEEE Workshop on Robot and Human Communication*, pp. 11-17, 1991.
5. Yuichiro Anzai, “Human-robot-computer interaction in multiagent environment,” *Proceedings of the Fifth International Conference on Human-Computer Interaction*, Vol. 2, pp. 2-7, 1993.
6. Thomas K Landauer, “Let’s get real: A position paper on the role of cognitive psychology in the design of humanly useful and usable systems,” *Designing Interaction: Psychology at the human-computer interface*, J. M. Carroll (Ed.), pp. 60-73, Cambridge University Press, New York, 1991.
7. B. R. Woodley, C. Jennings, A. Conway, and S. M. Rock, “A Contestant in the 1995 Aerial Robotics Competition: Aerospace Robotics Laboratory Stanford University,” *Proceedings Manual, AUVS ’95 Technical Papers*, Washington DC, pp. 669-76, July 1995.
8. AUVSI Aerial Robotics Competition, http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html
9. S. M. Rock, E. W. Frew, H. L. Jones, E. A. LeMaster, B. R. Woodley, “Combined CDGPS and vision-based control of a small autonomous helicopter,” *Proceedings of the American Control Conference*, Philadelphia, 1998.
10. Andrew Conway, *Autonomous Control of an Unstable Helicopter Using Carrier Phase GPS Only*, Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 1995. Also published as Stanford SUDAAR 664.
11. Marc A. Ullman, *Experiments in Autonomous Navigation and Control of Multi-Manipulator, Free-Flying Space Robots*, Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 1993. Also published as Stanford SUDAAR 630.
12. H.D. Stevens, E.S. Miles, S. M. Rock, R. H. Cannon, “Object-Based Task-Level Control: A Hierarchical Control Architecture for Remote Operation of Space Robots,” *Proceedings of the AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space*, pp. 264-273, Houston, 1994.
13. Stanley A. Schneider, *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators*, Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 1989. Also published as Stanford SUDAAR 586.
14. Gerardo Pardo-Castellote, *Experimental Integration of Planning and Control for a Intelligent Manufacturing Workcell*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, 1989. Also published as Stanford SUDAA 675.
15. Howard H. Wang, *Experiments in Intervention Autonomous Underwater Vehicles*, Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 1996. Also published as Stanford SUDAAR 693.
16. Heidi Schubert and Jonathan P. How, “Space construction: an experimental testbed to develop enabling technologies,” *Telemanipulators and Telepresence Technologies IV*, pp. 179-188, Pittsburgh, 1997.
17. Don Norman, *Things that make us smart: Defending human attributes in the age of the machine*, Addison-Wesley, Reading, 1993.