Robotica (2000) volume 18, pp. 361–367. Printed in the United Kingdom © 2000 Cambridge University Press

Telerobotic Ground Control of a Free-Flying Space Camera*

Christoph W. Borst and Richard A. Volz

The Computer Science Department, Texas A&M University, H. R. Bright Bldg., College Station, Texas 77845-3112, (USA)
E-mail: cborst@cs.tamu.edu volz@cs.tamu.edu

(Received in Final Form: September 23, 1999)

SUMMARY
NASA has developed a free-flying camera known as AERCam to assist Space Shuttle and Space Station operations. The first AERCam vehicle was controlled from within the Space Shuttle using a hand-held controller. We have developed a system for controlling AERCam remotely from the ground, which involves significant communication delays. We have tested its use with NASA's AERCam dynamics simulation in place of the actual vehicle. Our ground system uses a predictive display[1] that is based on a similar simulation of Shuttle and AERCam dynamics. For both physical and numeric reasons, the behavior of the remote vehicle may drift from what is expected by the predictive display system. We have developed a mechanism to periodically re-establish correspondence of the ground simulation with the remote vehicle using tracking data that reports AERCam's recent behavior. Our system also provides a basis for future development of more advanced autonomous ground control for space vehicles such as AERCam.

KEYWORDS: Telerobotic control; Space camera; Ground simulation; Remote vehicle; Space shuttle; NASA

1. INTRODUCTION

AERCam is a free-flying camera developed by NASA's Johnson Space Center (JSC) to assist Space Shuttle and Space Station operations. AERCam transmits live video to an operator controlling it remotely from within the Shuttle. As part of an AERCam enhancement project, we have developed a system that allows a ground-based operator to remotely control AERCam. Significant communication delays are involved, and we have found that moderately frequent calibration of the ground system using data from a remote tracking system is necessary. We delivered and demonstrated our ground control system to JSC in August of 1997 using NASA's simulation of Shuttle and AERCam dynamics in place of the actual space vehicles.

AERCam will allow Shuttle crewmembers or ground personnel to monitor ongoing operations and perform visual inspections of exterior Shuttle or Space Station components without requiring extravehicular crew activity (EVA). This can reduce demands on crew time for these tasks as well as reduce risks associated with EVAs. Minimizing crew involvement in control of AERCam makes it possible to perform visual inspections that are otherwise too costly in terms of crew time. Appendix A contains a document that motivates our work by describing use of an advanced AERCam ground control system in an important real-world application.

Our system is designed to allow ground-based control of an early version of AERCam. JSC provided AERCam avionics software designed primarily for use with a system in which a Shuttle crewmember controls AERCam using a hand-held input device such as a spaceball. Our system allows a ground operator to control AERCam using a spaceball and provides a foundation for development of advanced ground control systems for future versions of AERCam. It can be extended to provide a higher level of autonomy (see Appendix A) and to work with attitude hold or motion planning avionics currently being developed by JSC.

As background, we begin by briefly describing the AERCam vehicle and past related work. This is followed by an overview of our system from an operator's perspective and a high-level view of the main system components. The implementation is then described in more detail, and we conclude with a discussion of lessons learned and possible goals for future work.

2. BACKGROUND

AERCam is a spherical vehicle with a diameter of 35 centimeters and a mass of 16 kilograms. It consists of a pair of video cameras and a headlight housed in a soft outer shell of Nomex felt. The current design includes one camera for a wide-angle view and another for a telephoto view, but future designs may carry stereo cameras. The first AERCam vehicle, pictured in Figure 1, was flown on December 3, 1997 during Shuttle Mission STS-87.

AERCam's motion is controlled by twelve nitrogen gas jets directed by the avionics software. The avionics software currently being developed at JSC for future AERCams is based on the 3T Architecture for Robot Intelligence.[2,3] It will consist of a set of small tasks, called "skills", that run periodically at a high frequency to control the thrusters. For example, one skill receives operator commands via radio modem, and another skill converts each command into thruster states to control the thrusters. JSC is developing a set of skills to add functions such as attitude hold, detection of hazards such as signal loss, and a prototype motion planning system. The role of the avionics software in our system will be detailed in a later section.

*This work was supported by NASA/JSC under NASA grant #NAG9–893

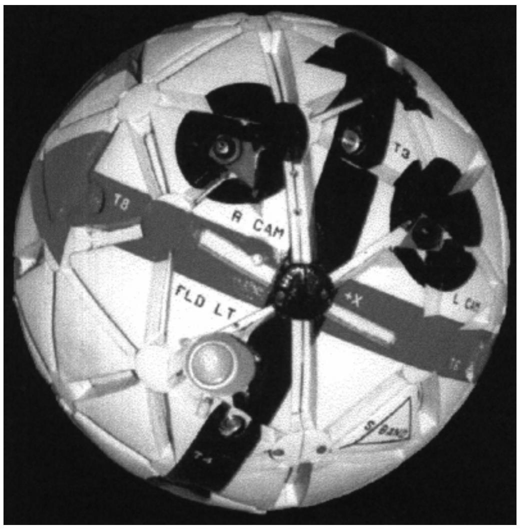

Fig. 1. AERCam I Free-Flyer

NASA estimates that ground-based control of AERCam will require the ability to deal with communication delays that are on the order of a few seconds in each direction. As part of the Universities Space Automation and Robotics Consortium (USARC), the robotics group at Texas A&M University has studied methods of dealing with communication delays in distributed telerobotic systems. In particular, we have developed simulation environments (e.g., TCS[4]) for visualizing remote robots using predictive displays[1,5] in which the operator sees both predicted and delayed behavior of the remote robot. USARC has also developed systems that allow multiple operators and autonomous systems at various locations to share control of a robot.[6] Our AERCam system has been designed to allow this type of shared control to be readily added in future work.

3. STRATEGY FOR AERCAM GROUND CONTROL

In our AERCam ground control system, a human operator interacts with a predictive display and uses a six-degree-of-freedom spaceball to control the vehicle's motion. The display shows the relative locations of AERCam and the Space Shuttle using animated 3D graphical models. The operator can select from various viewpoints and can have multiple views available simultaneously. In actual operation, the operator would also be able to view AERCam's video transmissions. This video view can be simulated by the graphical display.

To help the operator deal with large communication delays, the predictive display shows two graphical models of AERCam (see Figure 2). One is a smooth shaded model that shows the predicted AERCam position as calculated by a simulation in the ground system, and the other is a wireframe model that shows the last known position of the actual vehicle based on telemetry from a remote tracking system. The operator normally focuses on controlling the shaded model because it provides immediate feedback. Thus, the operator flies a predictive AERCam simulation using the spaceball, and the actual effect on the remote vehicle is reflected by the wireframe model some time later. If the remote vehicle's behavior is identical to that predicted by the simulation, then the wireframe follows exactly the same path as the shaded model. However, the remote vehicle's behavior cannot be predicted perfectly. Given enough time, even small errors can have a cumulative effect that renders the system useless unless they are dealt with. Much of our work has focused on constructing a mechanism to calibrate the predictive simulation from time to time based on information received about the remote vehicle's actual behavior.

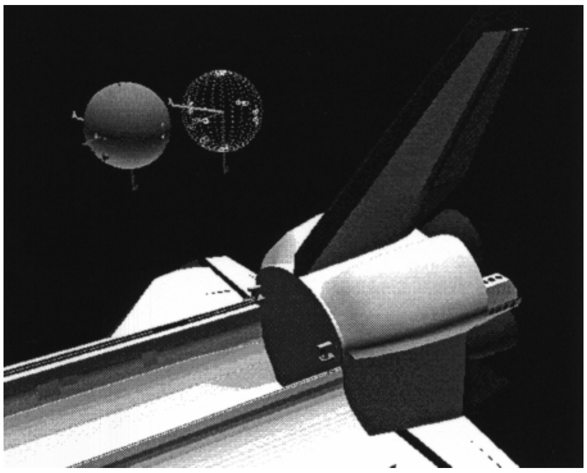

Fig. 2. AERCam with Shuttle in background

4. IMPLEMENTATION

The high-level view in Figure 3 shows the three main components of our system. The ground system communicates with a Shuttle-based system using the Information Sharing Protocol (ISP).[7] In ISP, information is sent to a server that forwards it to interested clients. This communications channel is where the large time delays occur, and we have included mechanisms for simulating these delays in our system. The ground system sends AERCam commands to the Shuttle-based system, which relays them to AERCam. The Shuttle-based system sends tracking data back to the ground using ISP. Tracking data and AERCam commands both include timestamps. Use of these timestamps will be described in later sections.

JSC's AERCam avionics software provides serial communication with a control station on the Shuttle using a UHF radio modem or communication over Unix sockets for

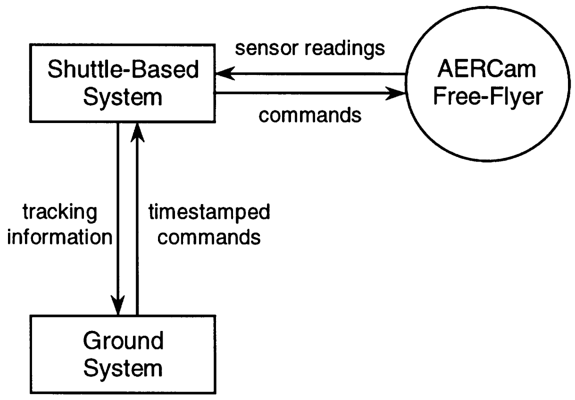

Fig. 3. System overview

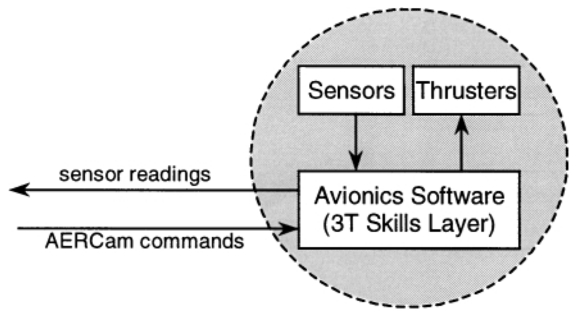

Fig. 4. AERCam Free-Flyer details

simulation without the radio modem. Our Shuttle-based system uses Unix sockets to communicate with an AERCam simulation in place of the actual vehicle. The simulated AERCam includes avionics software that receives the commands forwarded by the Shuttle-based system and transmits sensor readings or other status information back to the Shuttle-based system.

4.1 AERCam system and simulation details

Since AERCam was described only briefly in the background section, some additional details are given here. The relevant internal components are illustrated by Figure 4. AERCam's sensors consist of: (1) a set of accelerometers from which orientation and rotation rates are inferred, and (2) sensors for measuring temperature, fuel level, and battery power. The addition of position sensors or a tracking system based on referential GPS is currently being explored by JSC, but the details are not yet known. Our system includes a tracking module as described in Section 4.2.

In place of the actual AERCam vehicle, we use a dynamics simulation developed using the Trick Simulation Environment.[8] This simulation is a modified version of JSC's AERCam dynamics simulation. It is important to note that a reference frame attached to the Shuttle cannot be viewed as an inertial reference frame in which AERCam moves because the Shuttle rotates to maintain attitude with respect to the Earth and because of orbital dynamic effects (note that the two vehicles are in slightly different orbits). The Trick-based simulation includes full orbital dynamic effects for both vehicles. Variations of this simulation are used in three places in our system:

(i) in place of the actual vehicle, as described here,
(ii) for the predictive simulation (Section 4.3), and
(iii) for the recalibration mechanism (Section 4.4).

The AERCam dynamics simulation is controlled by the avionics software, which outputs changes in the twelve thruster states to the simulation. In return, the simulation provides accelerometer readings. For each of the twelve thrusters, the state is simply on or off, with no control over strength of the thrust. The avionics software derives the thruster states from motion commands it receives as input from the Shuttle-based system. Each command consists of six values corresponding to acceleration along six degrees of freedom. Each value is negative one, positive one, or zero to indicate negative, positive, or no acceleration along the

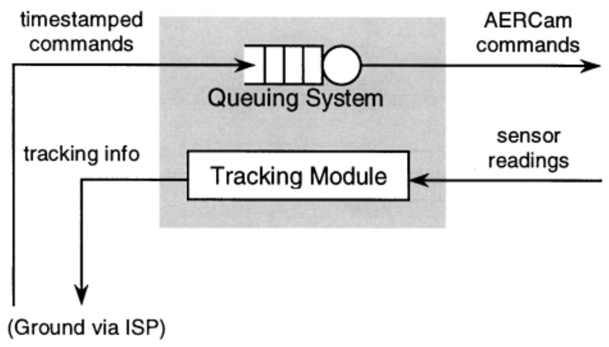

Fig. 5. Shuttle-based system details

corresponding degree of freedom. The avionics software we use only changes thruster states in response to incoming motion commands. However, future versions will implement minimum and maximum pulse widths that limit the duration for which a thruster can be activated per pulse.

4.2. Shuttle-based system details

The Shuttle-based system is shown in Figure 5. It consists of two main components: a system to track AERCam motion and a queuing mechanism that forwards commands from the ground to the AERCam vehicle. The details of the real tracking mechanism are still under development at JSC. Based on their advice, we have modeled a system for tracking position, orientation, velocity, and angular velocity as a module external to AERCam that periodically measures this information and sends it along with a timestamp to the ISP server for communication to the ground. Since the Shuttle is also moving, this module also returns corresponding measures for the Shuttle. Our tracking module obtains the needed measures directly from the dynamics simulation that runs in place of the actual vehicles.

The command uplink from the ground faces two timing problems: the obvious telemetry delays and a high variability in delay that can be injected by the ISP mechanism. The variability can cause significant deviation of AERCam's behavior from the ground prediction if care is not taken. To alleviate this problem, the timestamp of incoming commands is used. A queuing mechanism on the Shuttle forwards the commands coming from the ground to the AERCam vehicle at the appropriate time. This time is calculated by adding a constant offset to the timestamp associated with each incoming command. As a result, variations in the uplink delay are smoothed out, as long as the constant offset is greater than the maximum expected uplink delay, and assuming that the ground system and Shuttle-based system have synchronized clocks. For the remainder of this paper, the term 'uplink delay' will refer to this constant time difference.

4.3 Ground system details

The ground system components are shown in Figure 6. The spaceball module samples a spaceball's state at an operator-specified rate. When it detects a change in state, this module sends a motion command to both the predictive display system via a Unix socket and to the Shuttle-based system

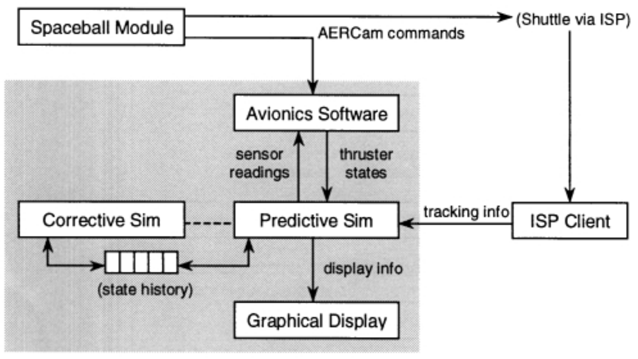

Fig. 6. Ground system details

via ISP. The motion command includes the timestamp used by the Shuttle-based queuing system.

The predictive display system on the ground consists of a copy of the avionics software, a simulation of AERCam and Shuttle dynamics, a 3D graphical display system, and a corrective mechanism for calibrating the predictive simulation based on incoming tracking data. The predictive display system's input consists of the motion commands generated by the spaceball module and the tracking data from the ISP server. The motion commands are received by the avionics software, which behaves similarly to the avionics software found in the remote AERCam system. The tracking data received via ISP is placed into shared memory that is accessible by both the predictive simulation and the corrective simulation described later. In addition to receiving tracking data, the ISP client simulates downlink delay by holding tracking data in a queue for a constant amount of time before placing it in shared memory. This simulated delay can be set by the operator and is added to any delay already caused by communication through the ISP server.

The predictive display system uses a graphical display tool called the Tree Display Manager (TDM[9]). TDM allows the operator to control viewpoints and open multiple views, and our environment provides a number of presets to choose from. Most of these provide viewpoints attached to the Shuttle's frame of reference to display a fixed Shuttle position, but some viewpoints are attached to the shaded and wireframe AERCam models to simulate the vehicle's camera view. A Trick-based dynamics simulation controls the graphical display by sending position and orientation updates for Shuttle and AERCam models to TDM.

At the heart of the predictive display system is a Trick-based simulation of full AERCam and Shuttle dynamics, much like the one that runs in place of the actual remote vehicles as described in Section 4.1. Here, the simulation predicts AERCam's future behavior and controls the TDM display. The simulation runs at a speed corresponding to real time. The shaded AERCam model is controlled using this predictive simulation, and the wireframe model is updated using position and orientation information from the remote tracking system.

If the simulation were to perfectly predict the remote vehicle's behavior, then the system described so far would be sufficient for a predictive display. However, due to a number of possible error sources, a correction mechanism is needed to keep the predictive simulation on track.

4.4 Corrective mechanism for predictive simulation

Error sources. There are a number of possible error sources that prevent the ground system's predictive simulation from exactly predicting future AERCam position over long periods of time. First, the predictive simulation may not perfectly model the dynamics of the vehicle because there are limits of accuracy for numerical integration of the equations of motion, and because all parameter values and their effects cannot be known exactly. This is complicated by the fact that the exact behavior of AERCam may change as thrusters get out of alignment or other changes occur.

During simulated operation, some of these effects can be emulated by changing parameter values in the predictive simulation or the remote AERCam vehicle simulation. Examples of such parameters include radius of orbit, mass of the Shuttle and AERCam, and strength of the AERCam thrusters. However, even if these parameters have identical values in both simulations, variations in behavior occur as a result of the fact that thruster states may not always change in the remote AERCam system at exactly the time that was used in the predictive simulation. Very small timing differences can be amplified by the fact that thruster state changes only take effect once per integration period in the dynamics simulations. For example, a thruster state maintained for approximately one integration period by the avionics software can effect zero, one, or two integration periods depending on the exact timing. This effect occurs in the predictive simulation during both real and simulated operation.

Our experience with the dynamics simulation, corroborated by JSC's analysis of the real device, is that the AERCam thrusters are very strong considering the vehicle's mass. As a result, even small variations in the timing of thruster state changes can have significant effects on AERCam behavior.

Even when these error sources only introduce small inaccuracies, the effects tend to be amplified over time. For example, a small error in angular velocity causes an increasing orientation error over time, and this in turn causes increasingly large errors in position as translations are commanded. In our experience, this can occur quite rapidly.

Correction concept. All of the errors described above are handled by a mechanism that corrects the current state of the predictive simulation while taking orbital dynamic effects into account. Our correction mechanism consists of an additional Trick-based simulation of AERCam dynamics that runs much faster than real time.

The main idea of the correction mechanism is to initialize a third (corrective) simulation to correspond to an earlier state of the predictive simulation, but with state corrections applied based on recent tracking data. Then, the corrective simulation is run as fast as possible (with the same input sequence already used by the predictive simulation) until it catches up with the predictive simulation. The resulting state of the corrective simulation is used to update the current state of the predictive simulation. Thus, the state of the predictive simulation is being re-calibrated based on knowledge about recent behavior of the actual remote system.

Corrective mechanism operation. To facilitate the correction, the main predictive simulation maintains a history of its state for each execution cycle, as seen in Figure 6. This history is kept in a circular buffer in shared memory, where it is accessed by both the predictive and corrective simulations. Each buffer entry includes: (i) simulated position, velocity, orientation, and angular velocity for the Shuttle and AERCam; (ii) information about thruster firings from the avionics software; and (iii) a timestamp.

The corrective mechanism runs periodically at an operator-specified rate. At the beginning of a correction process, the correction mechanism reads the latest available tracking data from shared memory and uses its timestamp to find the corresponding entry in the state history buffer. Since events in the remote system are delayed relative to their occurrence in the predictive system, the constant uplink delay must be subtracted from the tracking data's timestamp to find the appropriate history buffer entry. The buffer entry with the timestamp that most closely matches the result of this subtraction contains the state of the predictive simulation that corresponds most closely in time to the state of the remote AERCam system when the tracking measurement was taken. This history buffer state is updated using the tracking data and the resulting state is used to initialize the corrective simulation. Specifically, the position, orientation, velocity, and angular velocity are corrected for both AERCam and the Shuttle. After this correction is applied, the initial state of the corrective simulation represents a state that the predictive simulation should have been in several seconds before.

After initialization, the corrective simulation runs as fast as possible until it catches up with the main predictive simulation. In order to avoid any pause in operation of the predictive display, the corrective simulation runs concurrently with the main predictive simulation. As the corrective simulation runs, it uses thruster state data from the history buffer as input and updates the buffer states to reflect its correction at each execution cycle. The corrective simulation continues running until it overwrites the latest buffer entry written by the main predictive simulation. At that point, the two simulations enter a cleanup phase in which the corrective simulation calculates a final state to replace the current predictive simulation state. The main predictive simulation loads its corrected state and continues running, and the corrective simulation stops and waits until the next correction is needed.

The entire correction process is summarized below

(i) Initialize the corrective simulation:
   a. Subtract the uplink delay from the tracking data timestamp.
   b. Find the closest matching timestamp in the history buffer.
   c. Load state from the history buffer, with corrections applied based on tracking data.

(ii) Run the corrective simulation as fast as possible using the input sequence stored in the history.

(iii) Stop when caught up with the predictive simulation and copy back the resulting state.

The correction mechanism described here deals with errors in a general way that takes into account the orbital dynamics of AERCam and the Shuttle. Since the corrective simulation runs concurrently with the main predictive simulation, there is no pause in operation of the system.

5. OBSERVATIONS AND RECOMMENDATIONS

The predictive display system was run on an SGI Onyx with two MIPS R4400 processors, while the remote system was simulated on an SGI O2. The parameter values typically used for system evaluation were as follows:

Downlink delay:	3.0 seconds
Uplink delay:	4.0 seconds
Period for corrections to predictive sim:	20 seconds

With this configuration, the corrective simulation ran fast enough to perform its correction in less than a second. The correction mechanism was run periodically in our system, but future versions may perform corrections as needed based on current knowledge about error. Information in the history buffer of the ground system can be compared with incoming tracking data to determine when the past error in the system exceeds some threshold.

Given a reasonably accurate predictive simulation and tracking system, we found that the dominant source of error in the predictions can be attributed to timing variations in thruster state changes as described in Section 4.4. The strength of the vehicle's thrusters causes the effect to be quite significant. The operator can minimize the effect by avoiding sequences of several short thrusts whenever possible.

From the operator's perspective, the correction mechanism can result in a noticeable jerk in the display when the predicted position and orientation are updated if significant error has built up since the last correction. This jerk is most noticeable when the scene is being viewed from the predicted AERCam's point of view. The effect can be reduced by running the correction process very frequently to control error, but there may still be an occasional jerk as objects can appear to move substantially with small changes in the camera's orientation. To make corrections appear smoother to the operator, a gradual change in the display could be used. This could be accomplished by adding a time period to the correction during which both the predictive and corrective simulations run in synchronized fashion to calculate position and orientation information. The transition would then be made gradually by weighting the influence of each simulation on the display as a function of time rather than suddenly correcting the predictive state. This type of blending function has been used before in other systems for making smooth transitions between command modes.[6] It is possible, though, that a smooth transition could be more confusing to the operator than a jerk if it is perceived as motion of the vehicle.

Our ground system provides a framework for building more advanced AERCam ground control systems in the future. As more autonomy is added, new versions of the avionics software can be placed into the system. The system may require addition or modification of modules to deliver new types of AERCam commands to the avionics software. When a motion planner is used for AERCam, it may be necessary to modify our correction mechanism for the predictive display system since future position and orientation will already be known. Modules can be added to the system to integrate the ground control system with additional control stations on the Shuttle so that control of AERCam can be exchanged or shared among multiple operators or autonomous systems at various locations.

6. CONCLUSION

Ground-based control of AERCam makes it possible to perform frequent visual inspection tasks that are otherwise too costly in terms of Shuttle or Space Station crew time. We have built a ground control system for current AERCam technology using a predictive display based on dynamic simulation. The predictive system accumulates error over time due to limits of accuracy with which the remote system can be simulated, and the use of strong thrusters in AERCam causes this to occur quite rapidly. For this reason, our predictive display system includes a correction mechanism that periodically corrects the simulation state based on incoming data about AERCam's past behavior. Our correction mechanism takes into account the orbital dynamics of the remote system and does not result in a pause in operation.

In summary, we have demonstrated that ground control of AERCam is not only feasible, but readily accomplished in a manner that accommodates evolution toward increasingly autonomous operation.

References

1. M. Noyes and T. Sheridan. *A Novel Predictor for Telemanipulation Through a Time-delay* (NASA Ames Res. Center, Moffett Field, CA, 1984).
2. R.P. Bonasso and D. Kortenkamp, "Characterizing an Architecture for Intelligent, Reactive Agents," *Working Notes: 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents* (March, 1995), pp. 29–34.
3 R.P. Bonasso, D. Kortenkamp, D. Miller, and M. Slack, "Experiences with an Architecture for Intelligent, Reactive Agents," *Intelligent Agents II: Agent Theories, Architectures, and Languages*, (Springer-Verlag, Berlin, 1995), pp. 187–202.
4. M. Skubic, G.V. Kondraske, J.D. Wise, G.J. Khoury, R.A. Volz, and S. Askew, "A Telerobotics Construction Set with Integrated Performance Analysis," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA. (August 5–9, 1995), **Vol. 3,** pp. 20–26.
5. L. Conway, R.A. Volz, and M. Walker "Tele-Autonomous Systems: New Methods for Projecting and Coordinating Intelligent Action at a Distance," *IEEE Transactions on Robotics and Automation*, **6,** 2, 146–158 (April, 1990).
6. S. Graves and R.A. Volz "Action Selection in Teleautonomous Systems," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA. (August, 5–9, 1995), **Vol. 3**, pp. 14–19.
7. M.R. Barry, K.L. Scott and S.P. Weismuller, "A Distributed Computing Model for Telemetry Data Processing," *J. Robotics and Computer-Integrated Manufacturing* **11** (2), 99–104 (1994).
8. R.W. Bailey and E.J. Paddock, *Product Specification of the Trick Simulation Environment*, (LinCom Corporation TM–020–4630–96, March, 1996).
9. R. Strong, K. Mancus, R. Decker, and M. Voss, *TDM Developer's Guide*, (McDonnell Douglas Space Systems Houston Division TM–6.24.62–13, May, 1993).

APPENDIX A

Ground-Based Inspection of ISS (excerpt from Report to JSC)

Christoph W. Borst and Richard A. Volz

1. Introduction

When the International Space Station (ISS) begins operating, high demand will be placed on its crew to perform a wide range of tasks. These include assembly of the Station, housekeeping, monitoring of various systems, execution of numerous experiments in payload modules, and tasks related to exchange of cargo with the Space Shuttle. Experiments will already be underway early in the ISS assembly phase. At that time, the Station will initially be occupied by a crew of three, but will later house a crew of six along with dozens of ongoing experiments. The amount of crew time available for additional tasks is therefore very limited, and it is desirable to automate as many tasks as possible.

One specific area worthy of consideration is the task of inspecting exterior Space Station components. These components will be susceptible to damage from collisions with debris or micro-meteoroids. NASA has designed whipple shields to offer some protection from debris damage. In particular, small particles. (approximately one centimeter or less in diameter) will be broken apart and scattered by the shields to minimize damage to ISS. It will be useful to periodically inspect the whipple shields themselves, both to detect serious damage to them and to gather data about frequency and size of debris hits. Other ISS components, such as the photovoltaic arrays, will also need to be inspected for similar damage. The inspection procedure will be costly in terms of crew time and will limit the amount of crew time available for other tasks. As a result, the inspection is sure to be done infrequently. By automating the procedure and controlling it from a ground station, demands on crew time would be reduced, allowing inspections to be carried out frequently. In addition to determining damage, adequate inspection frequency would allow validation of NASA's debris models for the ISS.

NASA's AERCam project is developing a robotic, free-flying camera that continuously transmits video. The current design includes emerging technology for telerobotic control of AERCam from within the Space Shuttle or ISS, allowing visual inspection of external components without EVA. However, current ground operation is limited to the display and recording of information. The current technology is the basis for adding ground control that can extend its utility.

2. AERCam ground control system

With ground control added, it will be feasible to use AERCam to perform thorough inspections of ISS components that are susceptible to damage from debris at an adequate frequency. A ground station interface will display both the live video transmission from AERCam and graphical models of AERCam and ISS to show their relative positions. The relative position information used will be based on telemetry received from a tracking system onboard ISS. Two possible choices for this system are: (1) referential GPS or (2) a radar-based tracking system. The ground station display will also show a predictive AERCam location to help operators deal with time delays encountered in remote space operations.

Remote operators will have access to commands that correspond to different levels of autonomy in AERCam. At the lowest level, an operator will be able to manually control ("fly") AERCam using a hand-held controller. At a higher level, operators will be able to give target-centered motion commands by graphically providing a target position to a path planning module that will generate a path and pass it to an autonomous controller that will "fly" AERCam to the target position. As AERCam moves through a series of inspection points, images of the ISS will be taken and compared with previous images. If needed, the ground station interface will allow remote operators to rehearse these commands and preview simulated effects before commanding AERCam to perform the actual operations.

Ultimately, the inspection process may benefit from a higher level of automation. Operators may use high-level commands such as "inspect whipple shield." Two levels of automation may ultimately be possible: (1) planning the sequence of inspection points to fully inspect the area under consideration and (2) automated comparison of a current image with a previous image. The operator would supervise or monitor the system as it plans and executes a sequence of AERCam commands to accomplish its goal. When the system detects something of possible interest during inspection, it would pause the inspection and request additional input from a human operator. Similarly, an operator may pause the inspection if something interesting is seen during monitoring of video transmissions. Then, a manual control mode or other commands will be available for performing a more detailed inspection of the area of interest.

ISS crew time will be needed only for release and retrieval of AERCam and when the inspection process discovers damage requiring further attention from the crew.