

Integrated Planning for Telepresence with Time Delays

Mark D. Johnston and Kenneth J. Rabe

Jet Propulsion Laboratory, California Institute of Technology

4800 Oak Grove Drive, Pasadena CA 91109

{mark.d.johnston,kenneth.rabe}@jpl.nasa.gov

Abstract

Teleoperation of remote robotic systems over time delays in the range of 2-10 seconds poses a unique set of challenges. In the context of a supervisory control system for the JSC Robonaut humanoid robot, we have developed an “intelligent assistant” that integrates an Artificial Intelligence planner (JSHOP2) with execution monitoring of the state of both the human supervisor and the remote robot. The assistant reasons simultaneously about the world state on both sides of the time delay, which represents a novel application of this technology. The purpose of the assistant is to provide advice to the human supervisor about current and future activities, derived from a sequence of high-level goals to be achieved. To do this, the assistant must simultaneously monitor and react to various data sources, including actions taken by the supervisor who is issuing commands to the robot (e.g. with a data glove), actions taken by the robot, and the environment of the robot, both as currently perceived over the time delay, along with the current sequence of goals. We have developed a “leader/follower” software architecture to handle the dual time-shifted streams of execution feedback. In this paper we describe the integrated planner and its executive, and how it operates in normal and anomaly situations.

1. Introduction

Teleoperation of humanoid robotics with medium-range time delays (e.g. 2-10s) presents a variety of challenges of direct relevance to NASA’s current exploration initiatives. Such systems are applicable to a wide range of tasks, from earth orbit to the lunar surface, and hold out a promise of safer and cost-effective operations. As part of a project to demonstrate the combined use of supervisory control techniques, operator intent prediction, and an immersion cockpit for monitoring and control of robot state, we have developed an intelligent Task Level Assistant (TLA) that integrates an Artificial Intelligence planner with execution monitoring of the state of both the human supervisor and the remote

robot. In this mode the planner must reason about the world state on *both sides of the time delay*, which represents a novel application of this technology. The application domain of our system is the JSC Robonaut humanoid robot. The purpose of the assistant is to provide advice to the human supervisor about current and future activities, derived from a sequence of high-level goals to be achieved. To do this, the assistant must simultaneously monitor and react to various data sources, including:

- actions taken by the supervisor who is issuing commands to the robot (e.g. with a data glove), and actions taken by the robot as reported over the time delay
- the environment of the robot, as currently perceived over the time delay
- the current sequence of goals

As any of these change, the assistant must respond appropriately, detecting normal task completion as well as exception conditions.

A number of methods have been developed to help deal with teleoperation of remote robots across a significant time delay: see Sheridan [1, 2] for an extensive review. The technique used in our system is *supervisory control*, in which the remote robot has some degree of autonomous capability, and higher level commands are sent by the human supervisor. Supervisory control thus includes elements of autonomy and intelligence, although the intelligence is usually associated only with the human supervisor. Other techniques that emphasize machine intelligence include mixed-initiative control [3] and adjustable autonomy [4]. A key goal of these approaches is to shift more responsibility from the human to the machine.

In the following, we first describe the overall context of the system (Section 2) and the software architecture of the Task Level Assistant (Section 3). We then describe the planning component in more detail and the requirements derived from its use in a telepresence environment under active control by a human supervisor (Section 4). The time delay issue is addressed in Section 5, including the strategy of having the planner and associated execution monitor make use

of a “leader/follower” model. This section also discusses the implications of unexpected events, including the reaction to failure and the intentional or unintentional deviation of the supervisor from the high-level goal sequence. Also included in this section is a scenario to illustrate the behavior of the system in a characteristic reactive situation. We summarize our conclusions, status, and directions for further research in Section 6.

2. System Overview

Our domain [5] consists of a smart cockpit and a remote robot, each on separate sides of the time delay. The robot is the Robonaut [6] anthropomorphic humanoid robot (Fig. 1), developed at NASA’s Johnson Space Center specifically for space operations. The robots have over 40 DOF, with two 7-DOF arms with 5-fingered hands, and pan/tilt stereo vision cameras. The Robonaut software provides a number of autonomous primitive behaviors (e.g. move to touch, grasp to position or force, etc.) that can be commanded by the supervisor.



Figure 1. The NASA Robonaut dexterous robot

The human supervisor works in a smart cockpit environment, which includes the hardware and software required to support remote operations. There are video and monitor displays of the remote (time-delayed) robot. The supervisor has virtual reality (VR) immersion equipment as well, including a VR helmet along with a data glove for commanding the robot. The software in the cockpit includes an Operator Intent Prediction [7] component, an immersive environment for the supervisor [8], and the subject of this paper, the

Task Level Assistant (TLA), which plans and monitors tasks performed by the supervisor and the robot.

The TLA’s fundamental role is to generate an advisory task list (plan) based on a higher level sequence of goals, and then monitor execution of those tasks, advising the supervisor of progress towards achieving the goals, as well as on deviations from the plan and potential corrective actions. A key feature of the TLA is its advisory nature: the human supervisor remains fully in control of robot commanding. The TLA must therefore present a stable and accurate view of the plan status, and of the next steps that the supervisor is advised to take to achieve the goals in the sequence.

The Robonaut environment for the system described here is a simple one: there are vertical and horizontal handrails that are to be grasped, picked up, and moved to a storage box. There is also a “button” to be pressed. These represent prototypes of activities that a robot might be designed to accomplish in a space or lunar construction task.

3. Task Level Assistant (TLA) Architecture

The overall architecture of the TLA in the context of the smart cockpit environment is illustrated in Fig. 2. Communication with the rest of the system is via a message bus over which messages and data are transferred. From this source comes status updates on those tasks that have been accomplished by the human supervisor (and thus which are translated into commands for the robot), and on those tasks accomplished by the robot (as received over the time delay). Also externally generated and managed is a goal sequence representing the intention of the supervisor. These goals may of course change at any time, and goals may be added or deleted.

At the core of the TLA is the planner component (Fig. 2 center), described in detail in Section 4. The role of the planner is to generate a detailed task list from the goal sequence, for presentation to the supervisor. In addition to the goal sequence, the planner requires two other major inputs:

- a *domain model*, describing the robot environment (handrails, buttons, etc.) and the operations of which the robot is capable (pick up handrail, push button, etc.)
- current knowledge of the state of the supervisor and robot

Based on these inputs, the planner converts the goal sequence and state information into a set of tasks to present to the supervisor, which will accomplish the goal sequence starting at the current state.

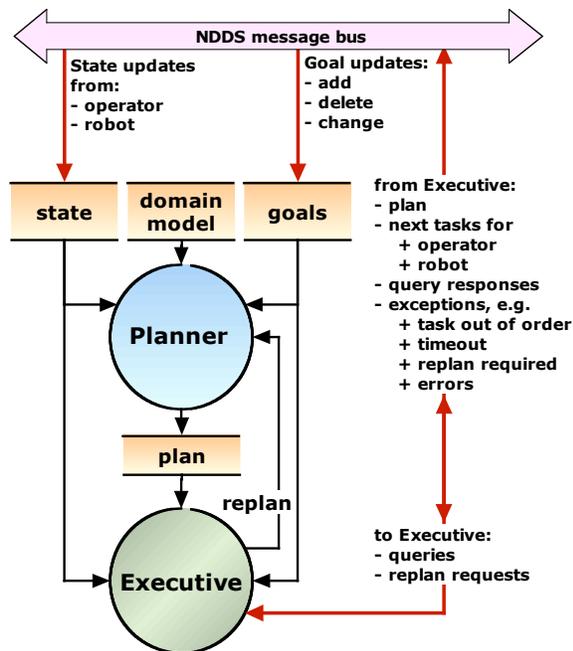


Figure 2. Architecture diagram of the TLA

The executive component (Fig 2 bottom) plays a crucial role in positioning the planner in an effective advisory capacity. It conveys the plan to the rest of the system. It monitors all changes from the rest of the system, including state information from the supervisor and robot, as well as goals. It correlates these changes with the current plan, and detects situations where a replan is required, then sends out a notification to this effect. In this application it does not automatically replan: it is specifically left up to the supervisor to request a replan (however such an automatic replan function would be straightforward to provide). In addition to “replan required” notification, the executive also identifies other situations of interest to the human supervisor, including:

- tasks executing out of plan order, either by the supervisor or by the robot
- remote robot timeouts, where task completion was expected to have occurred within some adjustable time limit, but has not been confirmed
- replan started/completed events
- notification that no feasible plan can be found

The executive also handles queries from the rest of the system, such as: next task to execute, execution status of a task, goal completion status, etc.

Finally, the executive handles replan requests from the supervisor, ensuring that publication of the current plan is placed on hold while a newly generated plan is being stored.

The implementation of the TLA is based on the Ensemble framework, an open architecture under development at JPL for mission operations software [9]. Ensemble is an adaptation of the Eclipse Rich Client Platform [10], and supports a component based application development model. The TLA has been developed in Java as a set of plug-in modules for Ensemble.

4. Integrated Planning

The planner chosen for the TLA is JSHOP2, a Hierarchical Task Network (HTN) planner developed at the University of Maryland [11] and presently available as an open source project [12]. The choice of JSHOP2 was driven by several considerations:

- HTN planners are complete (guaranteed to find a plan if one exists), and build up a plan in time-execution order, which is a natural way for the human users of the TLA to think about plan generation, i.e. based on planning a sequence of defined goals.
- JSHOP2 has a straightforward but comprehensive domain modeling capability, amply rich for the TLA Robonaut environment problem. HTN “methods” expand down into primitive operations with defined preconditions and effects on world state. As such, they are relatively straightforward to define by someone familiar with the domain.
- JSHOP2 is written in Java and was therefore relatively straightforward to integrate with the Eclipse Java-based platform chosen for TLA implementation.

JSHOP2 does not include built-in temporal reasoning, although it is possible to model a variety of temporal constraints by using appropriate variables. However, for this problem domain, temporal reasoning is much less of an issue than state and operation modeling, and so is not a serious drawback.

For this domain, the JSHOP2 model consists of the following:

- the two arms of the Robonaut robot, each with an availability state, and a “preferred” arm to be used if available
- a pair of handrails, one horizontal and one vertical, that can be grasped by the robot, moved, and placed in storage box
- a pushbutton that can be pressed by either arm

There are five primitive operations in the model, for picking up a handrail, dropping a handrail into the box, moving the arm over the box, setting down a handrail (not in the box), and pushing the button. There are two compound methods in the model: one for moving a handrail to the box, and one for pushing the button.

The move-rail-to-box method is shown in Fig. 3. Variables such as *?arm* and *?rail* will be bound to concrete values during the plan search process.

```

; move a rail to the box
(:method (move-rail-to-box ?arm ?rail ?goal)

  all-done
  (and (accomplished ?goal))
  ()

  arm-holding-rail
  (and (arm-available ?arm1) (holding ?arm1 ?rail)
        (not (accomplished ?goal)))
  ((!move-to-box ?arm1 ?rail ?goal)(!drop-in-box ?arm1 ?rail ?goal))

  arm-clear
  (and (arm-available ?arm) (clear ?arm) (not (in-box ?rail))
        (not (accomplished ?goal)))
  ((!pickup ?arm ?rail ?goal)
    (!move-to-box ?arm ?rail ?goal)
    (!drop-in-box ?arm ?rail ?goal))

  pref-arm-not-avail
  (and (arm-available ?arm1) (clear ?arm1) (not (in-box ?rail))
        (not (accomplished ?goal)))
  ((!pickup ?arm1 ?rail ?goal)
    (!move-to-box ?arm1 ?rail ?goal)
    (!drop-in-box ?arm1 ?rail ?goal))
)

```

Figure 3: An example TLA JSHOP2 method.

A JSHOP2 planning problem is defined by (a) a set of state values for all relevant variables, and (b) a sequence of goals to be achieved. The state values specify the assumed states of all entities at the start of the plan: e.g. that there are two handrails present, that both arms are available and not holding anything, etc. The goals are phrased as methods with specific values for their arguments, e.g. move rail “horiz- rail1” to the box with preferred arm “left”, to achieve goal “goal2”.

The result of plan generation is a list of ground operations, i.e. primitive operations with specific values for all variables. Because JSHOP2’s HTN planning algorithm is complete, it is guaranteed to find a plan if one exists. It may be than no plan exists (e.g. both arms are unavailable), in which case an empty operation list will be returned. Our experience has been that for the TLA Robonaut domain, JSHOP2 runs very quickly and is not a source of significant delay in the system. Of course, for large models where more extensive search is required, this could become an issue.

5. Leader/Follower Planning and Execution Monitoring

In a conventional planner/executive architecture, the preconditions of each task are checked before it is initiated. In a time-delayed environment, this reduces to inefficient “bump-and-wait” execution. Feedback from the human supervisor’s completion of a task is indeed nearly immediate, but feedback from the remote

robot is delayed, possibly by many seconds. To delay precondition checking and task dispatching is very inefficient, as illustrated in the schematic timelines of Fig. 4. Especially in the case where the time to perform operations (*T*) is relatively short compared to the time delay (*D*), it is wasteful to command an action and then wait for its completion before commanding the next one (Fig. 4a). Instead, it is desirable for the supervisor to be able to work ahead of the remote robot, thus realizing efficiencies analogous to “pipelining” commands (Fig. 4b). The efficiency gain can be substantial: waiting for remote tasks to complete lengthens overall execution time by the ratio *D/T*.

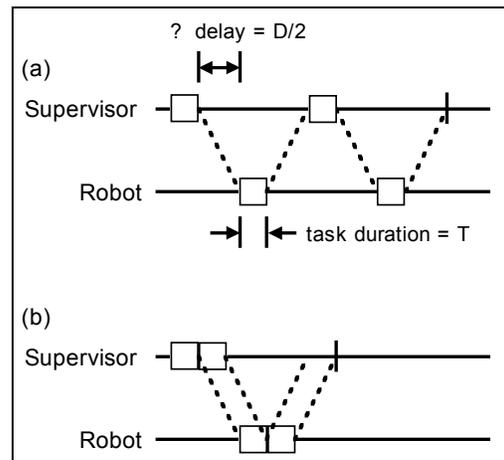


Figure 4. Time delayed task confirmation

Thus for TLA we were led to implicitly split the tasks that make up the plan into “leader” (human supervisor) and “follower” (time-delayed robot) stages. There are thus dual feedback paths into the executive (and thus back to the planner) from these split tasks: a task may be completed by the leader, thus downstream leader tasks can be dispatched and the leader can “work ahead” of the follower as in Fig. 4b. However, a task is not truly complete until it is also done by the “follower” (the remote robot). Provided this happens in a timely manner, the plan will progress on track. Indeed, it is essential for the planner/executive to operate on the normal assumption that the follower will indeed follow, until this assumption is violated. Violations may occur when the human supervisor deviates from the goal activities, either intentionally or unintentionally; the robot fails to execute a command as directed, or times out; or the state of the robot environment changes or is progressively revealed. In its role as assistant, the TLA must gracefully handle these situations and notify the human supervisor. To replan, the current state model must be “backed up” to the most recent known state of the robot. Replans must

also recognize which goals are accomplished and not expand and schedule their constituent tasks again.

The design principles we adopted for the leader/follower split were as follows:

- allow the leader to work arbitrarily far ahead of the follower, but keep track of expected time of completion of follower tasks, and provide notification if this time expires
- when replanning, fall back to the most recently confirmed state from the follower, i.e. “roll back” state changes that are due solely to leader task completion
- recognize any goals that have been accomplished and do not re-expand them when replanning
- ensure plan continuity, e.g. if a plan makes use of the left arm, but the preferred right arm comes back in service, plan to complete ongoing left arm activities before planning to use the right arm again

5.1. Normal Plan Execution

For testing purposes, we have implemented a zoomable graphical user interface for the TLA based

on the Piccolo toolkit [13][14]. A sample screen snapshot is shown in Fig. 5. The task sequence (ground operations and their arguments) is shown as the row of connected boxes, while the heavy vertical bar represents the break between the supervisor and the robot. Tasks to the *right* of the bar are the next ones for the human supervisor to execute. Tasks to the *left* of the bar have been executed by the human supervisor and are either confirmed completed by the robot (if grayed out) or are waiting for the robot to execute. If the next task for the robot has exceeded the expected time limit for confirmation, the alert box above it changes color. Below the task sequence is an event log, a current state snapshot, and testing controls for modifying the current state or the current goal sequence. In normal operation, the human supervisor can complete a set of tasks ahead of the robot and thus efficiently pipeline their combined activity. In Fig. 5, the supervisor has completed three tasks, while the robot has only acknowledged completing the first.

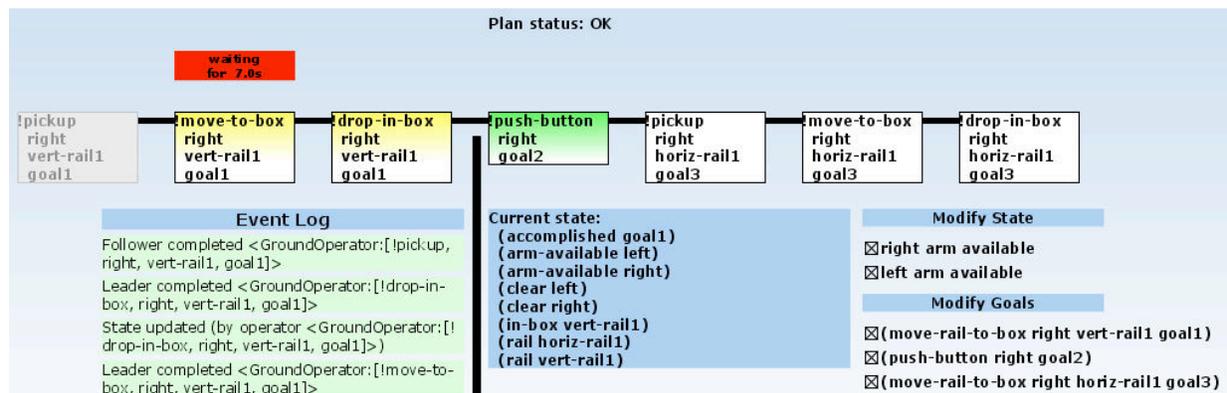


Figure 5: Normal planning scenario

5.2. Plan Deviations

There are a number of plan deviations that the TLA must handle gracefully. On the robot side, these include:

- *robot (follower) time out*: a task has taken longer than expected to confirm execution on the robot. In this case, TLA provides notification but takes no other action (in keeping with its advisory role)
- *out of order robot (follower) execution*: again, TLA detects and provides notification of this

situation, but takes no other action. Since this could be due not to failure to execute but simply to a communication gap, this is not treated by default by TLA as a reason to require replanning.

- *robot capability change*: this is a state change (other than produced by executing a planned task) that impacts the current plan, for example the loss of availability of an arm required for future tasks. In this case, TLA generates notification of the state change, and that a replan is required. Replanning is not automatic, as dictated by TLA’s advisor-only role.

On the human supervisor side, the operator can deviate from the planned task list, either intentionally or unintentionally. The approach taken by TLA in this situation is to *assume that the deviation is unintentional*, and that intentional deviations will be accompanied by goal changes to indicate the revised intent. Specifically:

- *human supervisor executes a task other than the next one in sequence*: TLA sends notification, and also that a replan is required. If a replan is requested at this point, TLA will generate a plan to accomplish the *original* goals, which may include additional tasks to return to the original plan.
- *human supervisor changes the goal sequence*: TLA sends notification and also that a replan is required. If a replan is requested at this point, it is based on current state and the revised goals.

In addition to plan deviations that fall into the above categories, other errors and exceptions are recognized and handled by the executive, e.g. completing a task twice, completing an undefined task, etc. In general, these lead to exception notifications and logging, but no other direct action.

5.3. Replanning

Replanning is done on request to the TLA from the human supervisor. TLA tracks situations where the current plan is compromised and a replan is necessary, but does not generate new plans automatically. This is to conform with its human advisory role, and the principle of “no surprises” for the human supervisor.

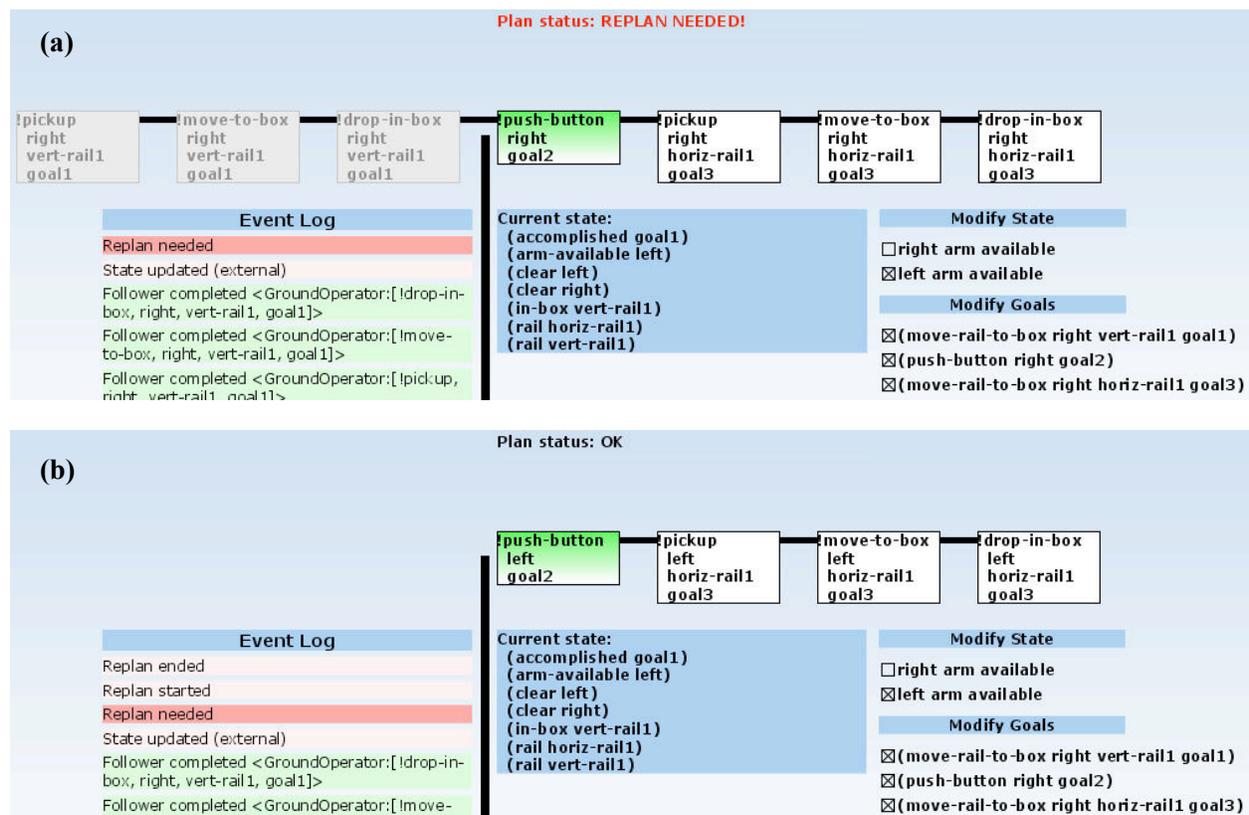


Figure 6: Replanning scenario due to change in right arm availability

When a replan is requested, TLA goes through the following steps:

- notify any listeners that a replan is beginning
- lock the plan against queries while the new plan is being generated

- roll back any state changes that were recorded due to the human supervisor (leader) completing activities and working ahead of the remote robot (follower).

- generate a new plan and replace the old one in memory, archiving any completed activities for future reference
- unlock for plan queries, and notify listeners that the replan is complete

An example replan scenario is illustrated in Fig 6. In this case, the supervisor and robot have executed the first goal of a plan with the right arm when it becomes unavailable (Fig. 6a). Following a replan, the remaining unaccomplished goals are planned for execution with the left arm, which remains available (Fig. 6b). The Event Log (lower left of each figure) lists routine transitions as well as notifications and situations in which a replan is detected as necessary.

6. Conclusions

We have described a novel approach to integrating planning with execution monitoring in the context of human supervisory control of a robot over a mid-range time delay. The Task Level Assistant (TLA) has been developed at JPL and run in a testbed environment to verify behavior in the Robonaut example domain. Our experiments to date have validated the choice of JSHOP2 as the core planner, and of the leader/follower architecture for tracking tasks over the time delay. JSHOP2 has provided fast, straightforward modeling and plan generation. The only drawback is the mapping to Lisp-like syntax for input, but we have automated this so it is transparent to users. It is worth noting that when the planner fails to find a plan, it is not always easy to determine exactly why. This is not a specific shortcoming of JSHOP2, however, but a reflection of the fact that interactions between plan model elements can be difficult to debug in general.

We expect to run TLA with the Robonaut hardware in 2006. The current simple domain model will be augmented to reflect a more complete model of states and tasks. Areas for future research and development include an automatically replanning version of the system that maintains a current “latest” plan as well as a stable “baseline” plan, and the incorporation of temporal constraints. We anticipate that the TLA may find application to other mission operations problems as well.

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

[1] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. Cambridge MA: MIT Press, 1992.

[2] T. B. Sheridan, "Space Teleoperation Through Time Delay: Review and Prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 592-606, 1993.

[3] J. A. Adams, P. Rani, and N. Sarkar, "Mixed Initiative Interaction and Robotic Systems," presented at 2004 AAAI Workshop on Supervisory Control of Learning and Adaptive Systems, Technical Report WS-04-08, 2004.

[4] D. Kortenkamp, D. Schreckenghost, and R. P. Bonasso, "Adjustable Control Autonomy for Manned Space Flight," presented at IEEE Aerospace Conference, 2000.

[5] K. Hambuchen, W. Bluethmann, M. Goza, R. Ambrose, K. Wheeler, and K. Rabe, "Towards Supervising Remote Dexterous Robots Across Time Delay," in preparation, 2006.

[6] W. Bluethmann, R. Ambrose, R. Diftler, S. Askew, E. Huber, M. Goza, F. Rehmark, C. Lovchik, and D. Magruder, "Robonaut: A Robot Designed to Work with Humans in Space," *Autonomous Robots*, vol. 14, pp. 179-198, 2003.

[7] K. R. Wheeler, R. Martin, V. SunSpiral, and M. Allan, "Predictive Interfaces for Long-Distance Teleoperation," presented at 8th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (ISAIRAS 2005), Munich, Germany, 2005.

[8] R. A. Peters, K. Hambuchen, D. Kawamura, and D. M. Wilkes, "The Sensory Ego-Sphere as a Short-Term Memory for Humanoid Robots," presented at 1st International Conference on Humanoid Robotics (Humanoids 2001), 2001.

[9] J. Norris, M. Powell, J. Fox, K. Rabe, and I.-H. Shu, "Science Operations Interfaces for Mars Surface Exploration," presented at IEEE SMC 2005, Hawaii, USA, 2005.

[10] Eclipse is described at <http://www.eclipse.org>

[11] D. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN Planning System," *Journal of AI Research*, vol. 20, pp. 379-404, 2003.

[12] SHOP and its variants are described at <http://www.cs.umd.edu/projects/shop>

[13] B. Bederson, J. Grosjean, and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Transactions on Software Engineering*, vol. 30, pp. 535-546, 2004.

[14] Piccolo is described at the site <http://www.cs.umd.edu/hcil/piccolo>