

PROVAR ASSISTIVE ROBOT INTERFACE

J.J. Wagner², H.F.M. Van der Loos¹, N. Smaby², K. Chang³, C. Burgar

¹Rehabilitation R&D Center, VA Palo Alto Health Care System;

²Dept. Mechanical Engineering, ³ Dept. Computer Science, Stanford University

Abstract

This technical paper describes the implementation of the User Interface for ProVAR, a desktop assistive rehabilitation robot. In addition to mediating interactions between the user and the real-time robot controller, the interface is responsible for the management of a world model and the high level validation of task deployment. The user may perform task planning, simulation and execution through a VRML and Java based 3D graphical representation of the workspace area and a Menu-bar command selection and edit window.

1. Introduction

The difficulty in placing assistive rehabilitation robots in the field is that both the end users and the occupational therapists who will train them are likely to have little or no previous experience with robots and possibly even limited experience with computers, erecting a high barrier to adoption and use. Thus the utility of an assistive robot is determined largely by the quality and ease of use of its User Interface (UI). The concepts and implementation discussed in this paper provide easier access to functionality than other robot interface concepts.

2. The Architecture

2.1 Pinocchio

Two workstations named Pinocchio and Jiminey,¹ are used in the ProVAR system, each under distinct division of labor and responsibilities.² The two computers communicate with each other via a secure, dedicated 100Mbit Ethernet connection and are protected from power failure by separate UPSs. Pinocchio serves as the controller for the Puma 260 robot. Pinocchio has a 200 MHz Pentium Pro running the QNX real-time operating system at a 500 Hz sampling rate, ensuring stable and safe robot behavior. In addition to controlling the arm, the main servo process can provide real-time simulation of the robot for use in the verification and confirmation of intended commands.

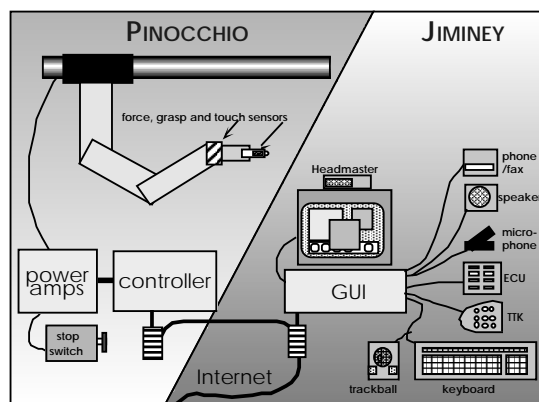


Fig. 1: ProVAR architecture

Pinocchio receives and processes single goal execution events that contain a command, a goal frame and a set of parameters. The servo loop is

able to provide synchronous limit and validity checking, e.g., joint velocity, torque and force limits. At any time, a new event may be given to preempt the current parameters. In addition, Pinocchio may be queried to supply current operating parameters, including joint angle, motor torque, and readings from force and proximity sensors³ located on the arm. The state table may be queried for either the actual robotic arm control or the real-time simulation.

Prentke-Romich (Wooster, OH) HeadMaster-Plus system for head motion cursor control, sip-and-puff and check operated switches, as well as standard keyboard and mouse/trackball inputs.

Jiminey has a high speed Ethernet connection to the Internet, and thus is not isolated from outside attempts to access it. Therefore Windows NT 4.0 was chosen as the operating system for the 266MHz Pentium II workstation for

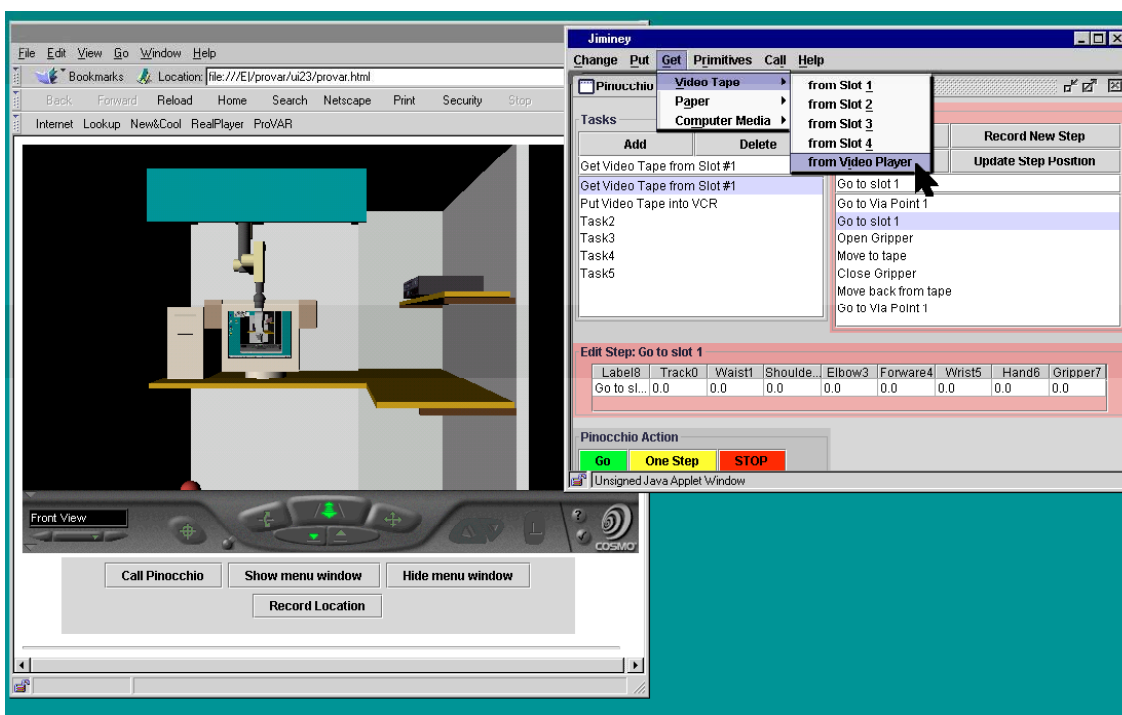


Fig. 2. The ProVAR VRML and Java based GUI

2.2 Jiminey

While Pinocchio is the real-time controller of the robot, the other workstation, Jiminey, handles the User Interface (UI) and performs high level task planning, management and execution. Communication between the user and Jiminey will be multi-modal, including a combination of user inputs via voice recognition software, a

the numerous security advantages it offers over Windows 95 and 98.

The ProVAR UI reflects the premise that the relationship between individuals and their assistive robots is fundamentally social. The ProVAR system is presented to the user as a “team” of two characters: Jiminey, a helpful consultant and Pinocchio, down-to-earth robot arm.⁴ While the

user perceives an engagement with both entities directly, in reality, the only direct interaction the user has with Pinocchio is the cheek-actuated emergency cut-off switch for the robot. All other communication with Pinocchio occurs through Jiminey.

In addition to mediating commands to Pinocchio from the user, Jiminey is responsible for the management of the world model and the high level validation of task deployment. For example, before executing a new task, the UI checks the task step list and world model for condition flags that need to be satisfied. When executing the task, shown in figure 2, i.e., “**Get Videotape from Video Player,**” Jiminey would first verify the state of several conditions:

- Is the last task/step complete?
- Is the gripper empty according to the world model?
- Is the gripper empty according to Pinocchio?
- Does the world show a tape in the VCR?

Other tasks performed by Jiminey include recording of full logging for the collection of data on real-time processes. This log is a valuable resource for debugging and field support of the system.⁵ The ability to extract the event history on the fly also increases the ability to perform remote maintenance and repair of the system, via ProVAR’s and ProVIP’s tele-diagnostic capabilities.

3. UI Design

The ProVAR UI is a VRML (Virtual Reality Modeling Language)

and Java-based GUI construct that affords the manipulation of ProVAR via examination of a 3D graphical representation of the world model and via a Java applet’s Menu-bar Command selection window. There are a number of VRML browsers available. Some are stand-alone applications but most, such as CosmoPlayer, run as plug-ins for web browsers such as Netscape or Microsoft’s Internet Explorer. The VRML and Java components are viewed while embedded in a web browser, creating a networked, platform-independent user interface for robot command. Support for voice recognition control uses the hooks in the Java menu bar for keyboard macros or, the Java Speech Application Programming Interface.⁶

In figure 2, the window on the left shows that the VRML robot can be moved around by cursor actions. The “Cosmo” controls along the bottom of the window allow the image to be zoomed. The gray buttons underneath transfer location data to/from the “Command-Edit” Java window in the right hand portion of figure 2. The individual steps of a command can be built and tested using the pull-down menus in the edit window.

4. The UI Components

4.1 VRML viewing of the world model

VRML is an object-oriented language, with a model typically consisting of the geometric description of an object with appearance and behavior nodes specified as needed. To create the ProVAR world model,

the work area and walls were created out of geometric primitives. Then a prototype of every “interesting” object (e.g., microwave, videotape, Puma 260) is created, then instantiated if and when it is needed.

In addition to creating static, three-dimensional representations, VRML supports animation through events sent to nodes via Javascript/VRMLscript and with Java applets.

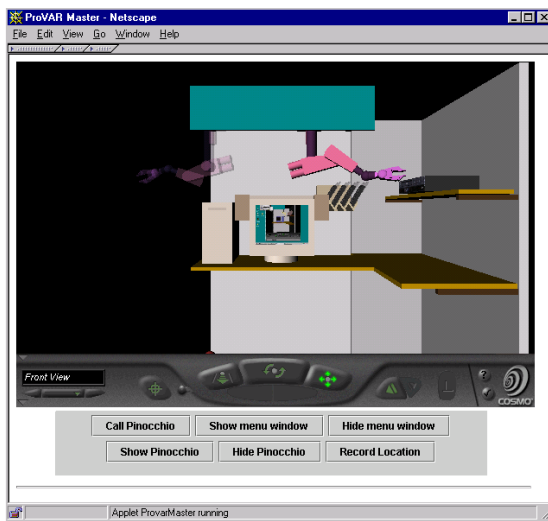


Fig. 3: Simulation robot (on right) and its marker ghost (on left).

There are two different colored robots that are viewable in the VRML browser window. One is the same color tan as the real robot (light gray in figure). The position in the VRML browser of the tan model always reflects where Pinocchio reports the Puma arm currently is (as seen in figure 2). The other Puma is colored a surreal magenta (dark gray in figure 3) and is used for simulation and testing of the next command to be sent to the robot. When the simulation robot’s

orientation is moved, it leaves behind a semi-transparent “ghost” marking its original position.

In addition to constructing or modifying the task to be sent to Pinocchio, in some instances, the manipulation of some objects in the VRML window can initiate real world activities such as operating an environmental control unit or answering the telephone.

4.2 Command-Edit window

Every command in the ProVAR system consists of a series of individual steps. These steps are grouped together to form tasks. A command is the completion of one or more tasks and may contain one or more branch points for selection between two different subtasks.. The Command-Edit window allows the user to create, simulate and verify all the steps and tasks in a command before sending them to the robot. For example, the command being created in figure 2 is “Play Movie...”

Tasks	Steps
Get Video tape from slot one	Go to Via Point 1
	Go to slot 1
	Open Gripper
	Move to Tape
	Close Gripper
	Move back from Tape
Put tape into VCR	Go to Via Point 1
	Go to VCR player
	.
	.

Table 1: Sample Command Task List

The steps in a Command list may be created and edited through a number of means. One of the easiest, if less accurate, methods is the direct manipulation of the VRML model into the desired configuration. Joint angles may be recorded and modified from the simulation of the robot, from encoder values taken from the actual robot that has been moved in position, or from keying in the numerical values directly into the joint position array. (see figure 4)

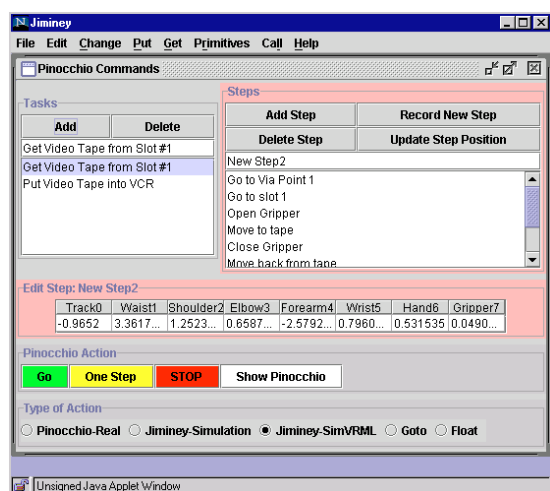


Fig. 4: Tasks can be built and tested using the pull-down menus in the Java applete Command-Edit window.

The Java task editing applet interacts with the VRML window via the EAI (External Authoring Interface). The EAI is a library of routines that allows a Java applet on a web page to access nodes and affect events and fields in a VRML browser embedded on the same page dynamically.

Previously created commands can be loaded and executed by selecting them using the menu bar of the Java applet.



Fig. 5: Example of cascading menu in a natural-speech suggestive format

References

- ¹ C. Collodi, *Avventure di Pinocchio. Giornale per i bambini*, Year 1, n. 1, July 7, 188
- ² Van der Loos, H.F.M., Wagner, J.J., Smaby, N., Chang, K.-S., Madrigal, O., Leifer, L.J., Khatib, O., ProVAR assistive robot system architecture, *Proceedings ICRA '99*, May 10-15, 1999, Detroit, MI pp. 741-746.
- ³ J.M. Vranish, Guiding robots with the help of Capaciflectors. *NASA Tech Briefs*, March, 1997, 44-48.
- ⁴ J.J. Wagner, H.F.M. Van der Loos, L.J. Leifer, Dual-character based user interface design for an assistive robot, *Proceedings ROMAN-98 Conference*, Kagawa, Japan, 9/30 – 10/2 , 1998.
- ⁵ H.F.M. Van der Loos. *A History List Design Methodology for Interactive Robots*. Ph.D. Thesis, Department of Mechanical Engineering, Stanford University, CA, 1992.
- ⁶ Java™ Speech API, Sun Microsystems, Inc., Palo Alto, CA, (<http://java.sun.com/products/java-media/speech/index.html>)