A-1

CONCURRENT SIMULATION OF A PARALLEL JAW END EFFECTOR

by

Bill Bynum
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185

ABSTRACT

This paper is an initial report on a system of programs developed to aid in the design and development of the command/response protocol between a parallel jaw end effector and the strategic planner program controlling it. The system executes concurrently with the LISP controlling program to generate a graphical image of the end effector that moves in approximately real time in response to commands sent from the controlling program. Concurrent execution of the simulation program is useful for revealing flaws in the communication command structure arising from the asynchronous nature of the message traffic between the end effector and the strategic planner. Software simulation helps to minimize the number of hardware changes necessary to the microprocessor driving the end effector because of changes in the communication protocol. The simulation of other actuator devices can be easily incorporated into the system of programs by using the underlying support that has been developed for the concurrent execution of the simulation process and the communication between it and the controlling program.

INTRODUCTION

One of the current research projects in the Automation and Technology Branch at the NASA/Langley Research Center on remote teleoperator control of robotic manipulators is the Distributed Artificially Intelligent System for Interacting with the Environment (DAISIE) [1,2,3], a system developed by Nancy Orlando of NASA that combines artificially intelligent, goal-structured robot planning algorithms with traditional control methods for mechanical equipment. In the DAISIE system, communication between the strategic planner and the various sensors and devices of the system is of paramount importance. When a new device is incorporated into the system, the set of commands that the device will accept and the responses that it will return must be specified, and then this command protocol has to be encoded in a form usable by the processor that controls the device. In the case of an end effector, the specification of the command protocol is straightforward, but conveying the protocol to the processor controlling the end effector consists of burning the communcation protocol program into the ROM of the processor. Unfortunately, it is impossible to forsee every eventuality when specifying a command protocol, and this leads to frequent modifications. Also, the subsequent incorporation of other devices into the DAISIE system could require changes in existing protocols. Modification of command protocols leads to the iteration of the previously described two-step process several times in developing a satisfactory command structure. The associated expense and effort of this process justify the use of software simulation to design and develop command protocols between the devices in the DAISIE system and the strategic planner.

The DAISIE system is housed in the Intelligent Systems Research Laboratory at NASA/Langley. The equipment in the laboratory includes a VAX11-750 (VMS), two PDP11-based workstations, two Unimate PUMA six-joint robotic manipulators equipped with either vacuum operated or parallel jaw end effectors, a vision system, and a data acquistion system. The equipment in the Lab is interconnected through an RTNET communication network and

can also be linked via analog channels to a CYBER 175 real-time control system. A more complete description of the Lab is contained in [1].

The graphics display of the end effector is generated on a DEC VS11 graphics terminal. Acceptable refresh rates are obtained by associated VS11 memory plane hardware, which overlaps storage of a future image in RAM with display of the current image. Movement of the end effector from one gap opening to another is portrayed by subdividing the total gap into smaller parts and showing the end effector in each of the intermediate positions. The speed of the movement is directly related to the number of subdivisions chosen and can be varied from barely perceptible to almost instantaneous. The user interacting with the LISP controlling program can be located at any terminal connected to the VAX and need not be using the VS11 terminal.

Interprocess communication and the simulation of concurrency is implemented through the use of the VAX/VMS system subroutines, accessible from FORTRAN [4]. These routines provide concurrently executing processes in the system the capability of communication through "mailboxes" and of synchronization through "event flags." Additionally, there is a system subroutine with which a process can "create" (or, initiate execution of) another process.

The strategic planner for the DAISIE system is being written in the version of LISP used on the ISRL VAX, Franz LISP. This LISP provides the facility to incorporate dynamically into the LISP environment separately compiled FORTRAN subroutines and functions, callable as LISP functions. This feature allows a LISP program to control the graphics, concurrency, and communications capabilites described above.

STRUCTURE OF THE SIMULATION SYSTEM

The simulation system is composed of three main parts:

- the controlling LISP program.
- a collection of FORTRAN subroutines imported by the LISP program to initiate execution of, and then to communicate with, the third part of the system.
- the FORTRAN program for the overall control of the communications and graphics subroutines which produce the simulation of the end effector.

To begin a simulation session, the user must load the file of LISP control functions into Franz LISP and initiate the master function at the top level. This function begins an initialization sequence that imports the necessary FORTRAN subroutines into the LISP environment.

One of these imported FORTRAN subroutines is called to start the simulation. This subroutine initializes the graphics display, the interprocess mailboxes, and event flag clusters, and then creates, as a separate process, the FORTRAN program that generates the graphics and simulates the end effector. Two mailboxes are created, one for commands sent from the controlling LISP program to the simulation program and the other for responses going the other way. Because VMS has a wait-for-offspring rule, this initialization subroutine executes to completion, but cannot terminate until the process that it created, the end effector simulation process, terminates. This ensures that the event flag cluster and the mailboxes that it creates continue to exist. Creating "permanent" mailboxes is another alternative, but the method chosen requires less privilege from VMS. At termination of the initialization subroutine, control returns to the calling LISP program, unlocking the keyboard and allowing additional LISP functions to be invoked.

The command and response information exchanged by the LISP strategic planner program and the actual end effector has the format of a five character field accompanied by a six-vector of real numbers. Each device in the DAISIE system communicates using this same

command/response format; this is one of the strengths of the design of the system. The character field gives the source and destination of the command or response, as well as the basic command or response. The six-vector of real numbers contains modifying information. For example, in the case of a move command, this information would include the type of move (ratchet, positional, etc.), and the end condition desired, along with any necessary parameters, such as rate gains, desired jaw gap or maximum jaw force at termination. In the case of a response to a status request, the six-vector contains the current and target jaw gaps, position and rate gains, and information from proximity and cross-fire detectors located on the end effector.

As a convenience to the user, each of the commands sent to the end effector simulator is embodied as a LISP function. This relieves the user of the burden of real-time parenthesis matching and of having to memorize the exact command formats and the encoding of their fields. Each of these LISP functions transforms its command into the DAISIE format and invokes one of the imported FORTRAN subroutines to place the command message into the command mailbox and set an associated event flag. The process simulating the end effector is waiting on this event flag. When the flag is set, the process retrieves the command from the mailbox and acts on it.

The move and initialize commands are enqueued by the simulation program for subsequent action, whereas the quit, status request, and debug commands receive an immediate reply. Commands involving movement of the end effector are enqueued because, in general, they require a longer time to complete than a status request. With this program structure, it is possible for the LISP driver program to issue a sequence of move commands and then to monitor their progress through a series of status requests.

When a command has been completed, the simulation process sends a reply to the LISP program by placing an appropriately formatted message in the reply mailbox and setting the associated event flag. At present, the simulation program assumes that all commands

complete successfully, although random failure reports could be easily introduced into the simulation program in the future. The quit command causes the release of the system mailboxes and event flag cluster, as well as the termination of the LISP driver function, the graphics program, and the initialization process that created it.

The debug command is not really a part of the DAISIE command structure, but was added to be able to obtain debugging information during the development of the simulation system. With it, the user can toggle the display of four different types of debugging information: event flag behavior, interprocess communication, end effector control, and graphics information.

DISCUSSION

This simulation system has been useful in revealing places where changes might be needed in the command structure. For example, since the response to the completion of a command, particularly a move command, can be widely separated in time from the command that was sent, it is difficult for the LISP program to associate the responses received with the command that was given. This indicates that perhaps the format of the DAISIE command should be expanded to include a sequence number to aid the LISP program in associating the response with the command sent.

It is a straightforward matter now to add simulators for the other devices used by the DAISIE system, such as force-torque or tactile sensors. The graphical representation of the state of device being simulated, although useful in the case of the end effector, is not essential to be able to incorporate the device into the simulation system for the purpose of developing and testing its command/response protocol. Incorporating additional devices into the simulation system will be useful in pinpointing situations where the responses from different devices present conflicting, ambiguous, or redundant information to the LISP constrolling program. It is anticipated that the capability of successively enlarging the collection of

command/reply protocols with which the LISP controlling program must deal will be a valuable tool in isolating unforseen problems in the command structure well in advance of the final coding for the strategic planner.

REFERENCES

1. Harrison, F.W. and Orlando, N.E., "A Systems-Level Approach to Automation Research", Proceedings of the 1984 Robotics Conference at the University of Alabama at Huntsville, Huntsville, Alabama, April 1984.

2. Orlando, N.E., "A System for Intelligent Teleoperator Research", presented at the AIAA Computers in Aerospace IV Conference, Hartford, Connecticut, October 1983.

3. Orlando. N.E., "An Intelligent Robotics Control Scheme", presented at the American Controls Conference, San Diego, California, June 1984.

4. VAX/VMS System Services Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts.