S33-61
78
188153

Systems Simulations Supporting NASA Telerobotics

F.W. Harrison, Jr. and J.E. Pennington
NASA Langley Research Center
Hampton, VA 23665-5225

ND 21049

1. Abstract

Two simulation and analysis environments have been developed to support telerobotics research at the Langley Research Center. One is a high-fidelity, nonreal-time, interactive model called ROBSIM, which combines user-generated models of workspace environment, robots, and loads into a working system and simulates the interaction among the system components. Models include user-specified actuator, sensor, and control parameters, as well as kinematic and dynamic characteristics. Kinematic, dynamic, and response analyses can be selected, with system configuration, task trajectories, and arm states displayed using computer graphics. The second environment is a real-time, manned Telerobotic Systems Simulation (TRSS) which uses the facilities of the Intelligent Systems Research Laboratory (ISRL). It utilizes a hierarchical structure of functionally distributed computers communicating over both parallel and high-speed serial data paths to enable studies of advanced telerobotic systems. Multiple processes perform motion planning, operator communications, forward and inverse kinematics, control/sensor fusion, and I/O processing while communicating via common memory. This paper describes both ROBSIM and TRSS, including their capability, status, and future plans. Also described is the architecture of ISRL and recent telerobotic system studies in ISRL.

2. Introduction

Historically, simulation has proven to be a cost-effective method for obtaining estimates of the feasibility and/or value of new technologies. This is particularly true where direct experience is difficult or impossible to obtain, either because physical systems do not exist and must therefore be modelled and simulated, or because access to the environment for testing is limited. Both conditions apply for space telerobotic systems and tasks such as in space assembly and servicing.

Two simulation and analysis environments have been developed to support telerobotics research. One is a high-fidelity, nonreal-time, interactive model called ROBSIM, which combines user-generated models of workspace environment, robots, and loads into a working system and simulates the interaction among the system components. Models include user-specified actuator, sensor, and control parameters, as well as kinematic and dynamic characteristics. Systems consist of up to five (5), ten (10) degree-of-freedom robot arms, on either independent or common moving bases. Kinematic, dynamic, and response analyses can be selected, with system configuration, task trajectories, and arm states displayed using computer graphics.

The second environment is a real-time, manned Telerobotic Systems Simulation (TRSS) which uses the facilities of the Intelligent Systems Research Laboratory (ISRL). It utilizes a hierarchical structure of functionally distributed computers communicating over both parallel and high-speed serial data paths to enable studies of advanced telerobotic systems. Multiple processes perform motion planning, operator communications, forward and inverse kinematics, control/sensor fusion, and I/O processing while communicating via common memory. Additional hardware elements of the simulation include symbolic processor, high-speed computer graphics system, manned control station, dual PUMA 560 manipulators, and a vision processor.

This paper describes ROBSIM, TRSS and ISRL, and their relationship to advanced telerobotic system studies performed at Langley.

3. High Fidelity Robotic Simulation

In 1981 Langley contracted with Martin Marietta Aerospace Corp., for an "Evaluation of Automated Decision-making Methodology and Development of an Integrated Robotic System Simulation." The objective was to bring within NASA machine intelligence methodologies in automated decision-making, and to develop a robotic system simulation as a testbed for applying this and other technologies needed for space robotic systems. The result was the identification of artificial intelligence (AI) methods applicable to automated decision making, and a framework for an integrated robotic simulation, including manipulator forward and inverse kinematics and dynamics [1].

Another result of the contract was the Remote Orbital Servicing System (ROSS) concept which, employing state-of-the-art technology, could service the Solar Max satellite, the Long Duration Exposure Facility, and the Space Telescope, to the same extent as man, in EVA or in the Shuttle payload bay [2]. Figure 1 shows the ROSS, in a two-arm configuration attached to a conceptual Orbital Maneuvering Vehicle. ROSS later became a focus for promoting space robotics within NASA [3].

Subsequent improvements to the simulation have included [4]:

1. Implementation of a preliminary interface between a computer-aided design (CAD) data base and the ROBSIM data base via the Initial Graphics Exchange Standard (IGES) format [5].

2. The ability to define the actuator and control system to the detail desired. For example, motor back emf and joint coulomb friction can be modelled, if desired. Either fixed-gain or adaptive control systems can be modelled, and response data can be output for analysis.

3. Extension of the simulation to incorporate multiple manipulator arms, each having up to 10 degrees of freedom, attached to either independent or common moving bases.

4. The capability for the user to quickly interactively define a simple arm geometry for conceptual design tasks, and then edit and refine the arm geometric and mass properties for detailed response simulations.

5. An interactive "help" feature which allows the user to interrupt the simulation, ask for and receive information on simulation characteristics and capabilities, and then resume processing.

6. Rehosting the graphics output to Digital Equipment Corp. (DEC) graphics-compatible terminals, enabling a low-cost graphics display capability.

ROBSIM runs on a DEC VAX 11/750 computer under the VMS operating system. Its speed depends upon the level of detail in the system definition and the subsystems simulated. When simulating the dynamics of multiple manipulator arms, performing path planning or detailed control system modelling, computing inverse kinematics for a kinematically redundant arm (having seven or more degrees of freedom), or when outputting to a low-performance graphics system, the simulation operates slower than real time, but still at an acceptable interactive rate. Figure 2 illustrates steps in using ROBSIM in a telerobotic system development cycle:

1. Interactive definition of manipulator arms(s) and end effectors, and base to which each arm is attached.

2. Definition of the static environment.

3. Definition of dynamic environment features, including the loads to be carried and manipulated.

4. Synthesis of the system.

5. Development of the task description, using a simple task language and elementary task functions.

6. Simulation of the task.

7. Graphics display of task plus performance data output to a printer, plotter, or terminal.

8. Analysis of results by the user and determination of desired changes.

Despite its capabilites and ease of use, ROBSIM has three significant limitations. First, it is essentially a rigid body simulation. Second, for most applications, it runs slower than real time. Third, it is primarily a robotic, rather than telerobotic simulation.

Several approaches to modelling manipulator structural flexibility have been examined including assuming rigid links and modelling bending as a compliant joint rotation, frequency domain analysis retaining lower order modes, and finite element analysis, and symbolic rather than numeric solution of the dynamic equations. One option being considered is to provide the user the option of interfacing to an existing finite element structural analysis program. This would enable the user to control the level of detail desired and provide an interface to computer-aided design and analysis programs.

A promising approach to speeding up dynamics calculations is to distribute the computation over multiple processors. The rigid body dynamics equations are recursive and coupled and appear suited to parallel processing. The application of concurrent processing to manipulator dynamics is an on-going research activity at Langley.

The distinction between a robotic system and a telerobotic system is important. With a robotic system the human can interrupt a task, but usually interacts with the system at the end of a task or process. A number of processes may occur simultaneously, with the human acting as a monitor or supervisor, but not as a controller. In a telerobotic system, the human can act as a monitor or supervisor of automated tasks, and can also share control with or take control from the robotic controller. Unlike ROBSIM, where the response of the simulated system may be output at rates faster or slower than real time, depending upon the complexity of the system-simulated, a telerobotic simulation must reference all timing to real time.

The ROBSIM program has been provided to over 20 organizations, including most NASA Centers, the U. S. Air Force, both aerospace and nonaerospace companies, and several universities. ROBSIM will continue to be improved and disseminated.

4. TeleRobotic System Simulation (TRSS)

In order to model and study telerobotic tasks, a real-time, man-in-the-loop TeleRobotic System Simulation (TRSS) was developed. TRSS enables investigations of space-related telerobotic applications through a combined hardware and software simulation utilizing the facilities of the Intelligent Systems Research Laboratory (ISRL). ISRL provides an environment where teleoperator and robotic technologies may be studied, and it enables integration and evaluation of telerobotic system and task hardware.

Development of TRSS began in 1981. In 1982 a computer graphics simulation of a 5 degree-of-freedom manipulator performing a precision alignment task, with time delays of up to 2 seconds, was conducted. The simulation ran on a Control Data Corp. Cyber 175 computer and used displays and controls in a general-purpose aircraft cockpit simulator. Results showed that subjects changed from continuous control to a move-and-wait strategy for time delays of .25 sec and longer. Time required to perform the simulated task increased linearly with time delay, but time delays had no effect on alignment accuracy [6].

In 1983, TRSS was rehosted to a VAX 11/750 in the Intelligent Systems Research Laboratory and interfaced to a control station, a general purpose data acquisition system, and a Unimate PUMA 560 manipulator. Subsequently, a vision system and a second PUMA 560 were added. The PUMA, typically used in "pick-and-place" industrial applications, is a digitally-controlled six degree-of-freedom (DOF) anthropomorphic manipulator. It has been augmented with a parallel jaw gripper and a six DOF wrist force/torque sensor. Each subsystem in ISRL uses an LSI 11-73 computer running the RT-11 operating system. Communication among processors occurs on a 250 Kbyte/sec packet-switching global bus conforming to IEEE Std. 488-1978. The VAX serves as network controller and common memory. A Hyperchannel interface provides a direct memory access (DMA) interface between the Cyber and VAX computers; a high-speed serial interface connects the VAX to a Symbolics 3670 computer; and a DMA interface is being implemented between the VAX and a Rediffusion Computer Graphics Corp. Poly 2000 high-speed computer graphics subsystem.

Figure 3 illustrates the hierarchical, distributed architecture in the current ISRL configuration. Figure 4 shows a similar architecture for the software modules in TRSS. The mapping of the TRSS modules was developed in a manner analogous to manned aircraft or spacecraft systems; the operator will be capable of modifying manipulator control strategies and assuming direct manual control at any time. However, repetitious or tedious tasks will be automated so the operator need only monitor or supervise these functions. The modularity of TRSS allows the user freedom to utilize as much or as little hardware as desired, and allows modification or functional replacement of any software module.

One of the first telerobotic studies in ISRL was an active compliance task. Strain gauges mounted in the fingers of the end effector sensed constraint forces and torques during close tolerance peg insertion, and fed this data to control and display modules via the data acquisition system. A simple graphics display (fig. 5) indicated the magnitude and direction of the binding forces and torques. Using the display the operator could readily command the arm to move to null any disturbing forces. TRSS was then modified to allow the operator to select a mode in which the strain gauge data were fed directly to the control system to automatically null the force and torques. The operator then used the display to monitor the automatic insertion.

The next step was to incorporate automated control based on vision sensing. The philosophy in TRSS is to partition vision processing between man and machine, giving each responsibility for that which it does best. Man performs image recognition and interpretation, while the machine vision system performs image acquisition, low-level processing, and determines the location of objects. In ISRL the image processing system involves a 16-level gray-scale imaging system, a 240 X 320 X 4 frame buffer, and processing algorithms on the PDP 11-73 computer. The simulation is structured such that only those modules which actively manipulate the vision system data structures (low-level image processing routines) will require modification if a different system is used. Current research efforts center on determining the three-space location and orientation of labeled objects using a single camera, and on a multifunction recognition operator for telerobotic vision [7].

In 1985, vision-based control and force/torque control were integrated in a simulated satellite servicing task. Figure 6 shows a mockup of a biostack experiment module carried by the Long Duration Exposure Facility (LDEF) satellite. Three bolts must be loosened before the simulated module could be removed. The operator's task was to position the television camera so as to acquire a labeled object in the field of view (in this case four LEDs). After acquisition the operator invoked the vision system, which determined the location of the module with respect to the camera and the end effector, and then commanded the manipulator to move to the module. After positioning the end effector above the first bolt, the vision system notified the operator and transferred control to the force/torque system, which moved the wrench to the bolt and commanded the end effector to torque the bolt for release.

An advantage of the TRSS control structure, which uses resolved motion commands to the manipulators, is it enables the operator to share control with the autonmous system rather than having to switch between manual and automatic control. Using joystick control (from switches, dual three-axis controllers, or six-axis controller) the operator can input motion commands in parallel with inputs from sensor-based automatic controllers.

During 1986, TRSS was enhanced to enable simultaneous control of both PUMA manipulators. Current studies are addressing dual-arm telerobotic assembly tasks. In order to manually control both arms simultaneously, task-referenced control options were implemented. This enables the operator to define a task reference point for translational and rotational control inputs, and then the manual commands are transformed and the manipulators driven, with respect to the reference point. For example, if a long truss member is grasped by both arms the task reference point could be selected between (or beyond) the two grasp points, and both arms would be commanded to move the truss member in a coordinated motion referenced to that point.

Despite the success in simultaneous control of two manipulators and in the teleoperator assembly of columns and nodes in ISRL using TRSS, a great deal of work remains to reliably automate space assembly tasks. For example, small inaccuracies in manipulator dimensions can cause disturbance forces when two arms try to move a column. Hybrid control methods based on force sensing have been proposed for this closed chain task, but have not yet been proven in TRSS. Multiple arm coordination and control is an active research area.

5. Virtual Architecture

Even simple telerobotic tasks require a large amount of software. In the traditional approach to telerobotic programming, the level of detail is minimized by encapsulating the most tedious programming into a real-time executive whose only user/programmer access is via a robot programming language. A serious limitation of this approach is the lack of portability. An alternative approach is one in which the most tedious programming still remains in the real-time executive, but applications programming is accomplished by manipulating well-defined data structures via the language of the user's choice. By doing so, manipulators, sensors, and controls are integrated in such a manner that conventional programming techniques (and conventional programmers) can be applied to telerobot programming. The idea is to provide a common reference model, or virtual architecture, for TRSS and other applications in ISRL.

A virtual architecture, designated the Teleoperator and Robotics Testbed (TART), is implemented on the VAX 11-750. Concurrency, mutual exclusion, and signaling mechanisms are provided by the VMS operating system. FORTRAN was chosen as the implementation language because its data structures are a subset of those present in most modern computer languages and it is the language familiar to the largest number of laboratory personnel. It should be emphasized that a program written in any language can be used for algorithm development so long as the TART data structures can be represented and manipulated. To date, languages used include FORTRAN, Pascal and C. The implementation suffers all the inefficiencies of high-level languages in general (and FORTRAN in particular); however, experience has shown these limitations are moderated by the additional flexibility and freedom offered by a general programming language.

TART is a layered product in which each successive layer provides additional value to the system. Currently, five layers are implemented: (1) user, (2) system, (3) scheduling, (4) communications, and (5) servo/sensor. The lowest four layers of TART are designed for minimal use of user resource quotas, but accomplish most of the detailed programming and error checking required by user applications. The algorithms used insure robust, device independent processing and control of sensor and actuator systems, and they allow user-layer applications to concentrate on algorithm development.

The lowest level of the TART system, the servo/sensor, contains processes to perform sensor interpretation and actuator control functions, Generally, the algorithms for these operations execute on dedicated controllers. An example of a servo/sensor process is the timer-driven routine implementing base-referenced Cartesian moves for a manipulator. This process is responsible for accepting and converting setpoint and trajectory information from user processes to torque commands for each manipulator joint. Health and status information is returned to user processes through the intermediate layers of the TART architecture.

To insulate higher levels from the peculiarities exhibited by specific hardware devices, the second layer of TART, dedicated to real-time communications and data conversion, is provided. This process examines the TART data structures and communications channels for state modifications. In the case where a change is detected in the data structures, a command is formed and transmitted to the appropriate hardware device. In the case where a communications channel requests service, the information is accepted and converted to a TART data representation.

The next system layer, real-time scheduling, is implemented as an asynchronous, demand-driven process. TART differs from most robot controller executives in that scheduling of most routines is accomplished implicitly through the TART data structures. Each user and system-defined instantiation of a data type contains a field specifying an evaluation function to be applied to the data. This field either is null, contains the starting virtual address of a subroutine local to the image, or contains the name of an entry point in an installed, sharable image [8]. The first case implies that the data is constant and no evaluation is performed. In the later case, the specified image is dynamically loaded into virtual memory before applying the evaluation function. The frequency of evaluation is specified in a timer field. Timer values are expressed in VMS format and may be null, delta, or absolute. A null entry implies immediate, one time evaluation; delta times imply periodic application of the evaluation function; and absolute times imply one time evaluation at some future time. This is similar to a mechanism for automatic update of matrices implemented by Hayward, but applied uniformly to all data types [9]. Scheduling is implemented via the VMS asynchronous system trap (AST) mechanism with the address of the data structure containing the entry address as the first argument of the AST argument list.

The fourth level of the architecture, the system level, provides mechanisms for the addition, deletion, replacement, allocation, and deallocation of user and system defined data structures to the simulation data base. For all TART applications, predefined system data structures are contained in an installed common memory, or mailbox, on the VAX in order to minimize overhead in data access. This mailbox is global to all processes running on the VAX and users may directly access system data structures when appropriate. Typical of system defined data structures is the structure which defines the PUMA manipulator contained in ISRL. Once a data structure is added to the simulation data base, maintenance of the structures and scheduling of its evaluation functions is done by the real-time scheduling layer.

Presently, the top layer of the system is designated the user level. While all TART data structures are global, users are encouraged to use only the TART-defined system level mechanisms for modification of data structures. Functions for most of the common data transformations required by robotics applications are provided at the user level as a library of subroutines. Templates and definitions for TART data structures may be included via language-specific constructs in any application.

6. Future Plans

Two future goals are to incorporate more ROBSIM capabilities into TRSS, and to improve the compatibility of ISRL and TRSS with other NASA telerobotic facilities. The TART architecture will support both objectives. Langley is supporting research in the application of concurrent processing techniques to improve the computation speed of the manipulator dynamics. Computing system dynamics at actuator update rates would increase the fidelity and performance of TRSS as well as reduce interactive delays when structural flexibility is modelled in ROBSIM.

To increase the level of compatibility of ISRL with the OAST Technology Demonstration program at JPL, the PDP-11 subsystem computers are being supplemented with DEC MicroVAX II computers. This upgrade will enable multitasking utilization of a high-speed serial communications bus as employed by JPL. The MicroVAX system will retain compatibility with the parallel bus structure presently used in ISRL, and enable evaluation of serial and parallel bus communications for telerobotic tasks.

7. Concluding Remarks

The objective of automation research at Langley Research Center is to advance technology in the areas of the mechanisms, controls, sensing, and operator interface required by space telerobotic tasks. Two of the principal tools used to accomplish this objective are a high-fidelity simulation of robotic systems (ROBSIM), and a real-time man-in-the-loop Teleoperator and Robotics System Simulation (TRSS). TRSS is based on a multi-level virtual architecture implemented in the Intelligent Systems Research Laboratory. Greater flexibility and reduced development time have been realized with this system architecture.

8. References

[1] J.W. Lowrie, A.J. Fermelia, D.C. Haley, K.D. Greban, J. Van Baalen, and R.W. Walsh, Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation. NASA Contractor Report CR 1659775, September 1982.

[2] Remote Orbital Servicing System (ROSS). Volume 1: Executive Summary. NASA Contract NAS1-16759, April 1982.

[3] A.J. Meintel, Jr., and R.L. Larsen, NASA Research in Teleoperation and Robotics. Society of Photo-Optical Instrumentation Engineers Conference, San Diego, CA, August 1982.

[4] D.C. Haley, B.J. Almand, M.M. Thomas, L.D. Krauze, K. D. Gremban, J. C. Sanborn, J. H. Kelly, and T.M. Depkovich, Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation. NASA Contractor Report CR 178050, March 1986.

[5] B.M. Smith, K.M. Brauner, P.R. Kennicott, M. Liewald, and J. Wellington, Initial Graphics Exchange Specification (IGES), Version 2.0. National Bureau of Standards NBSIR 82-2631 (AF), February 1983.

[6] J.E. Pennington, A Rate Controlled Teleoperator Task with Simulated Time Delays, NASA Langley Research Center, TM-85653, September 1983.

[7] P.W. Goode, A Multifunction Recognition Operator for Telerobotic Vision. AIAA Guidance, Navigation and Control Conference, Williamsburg, VA, August 1986.

[8] VAX/VMS Linker Reference Manual. Digital Equipment Corporation, Maynard, MA, September 1984.

[9] Advanced Industrial Robot Control Systems, Tenth Report, March 1, 1983 - September 1, 1984. Purdue University, West Lafayette, IN, July 1984.

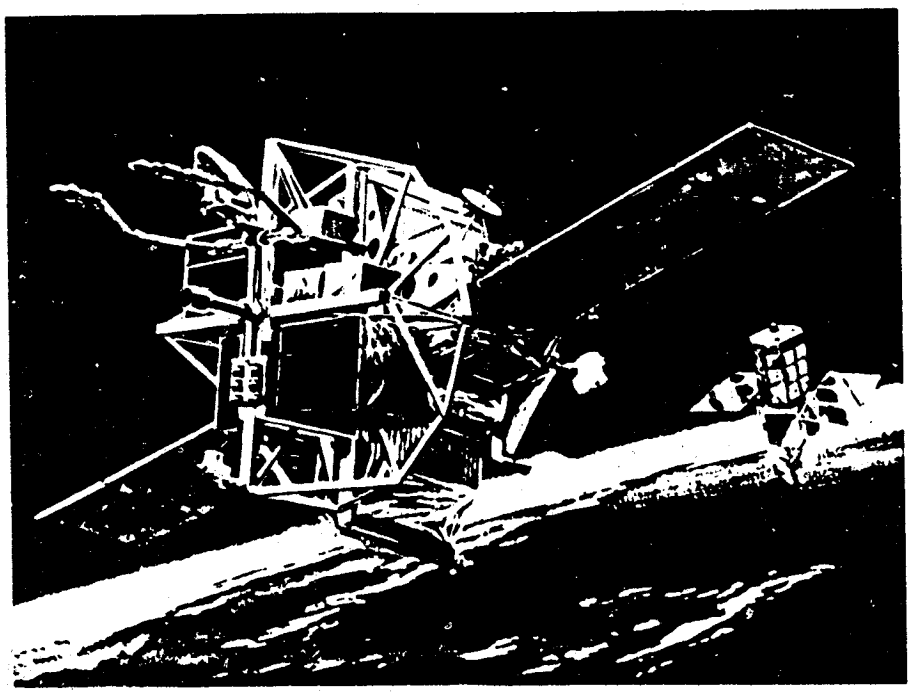


Figure 1. Remote Orbital Servicing System concept

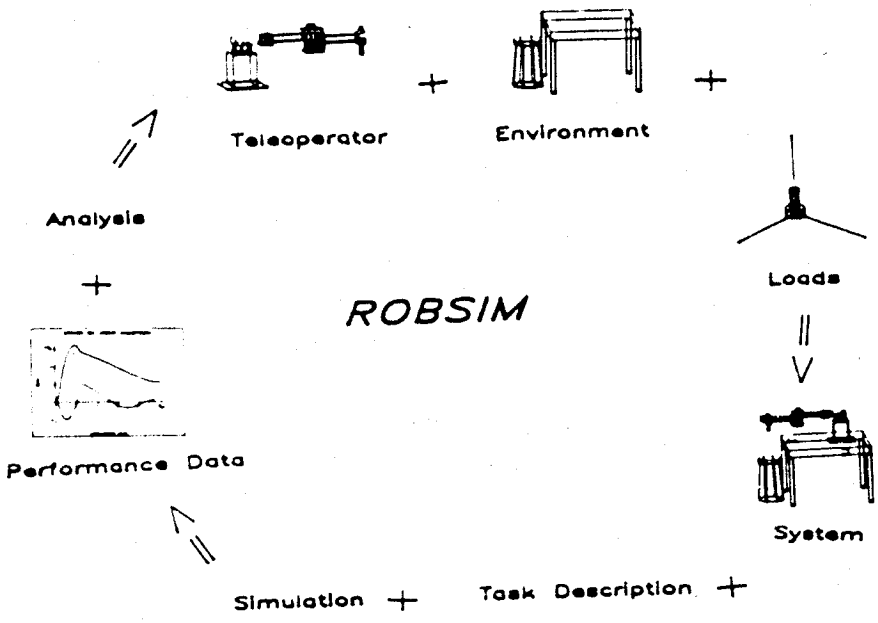
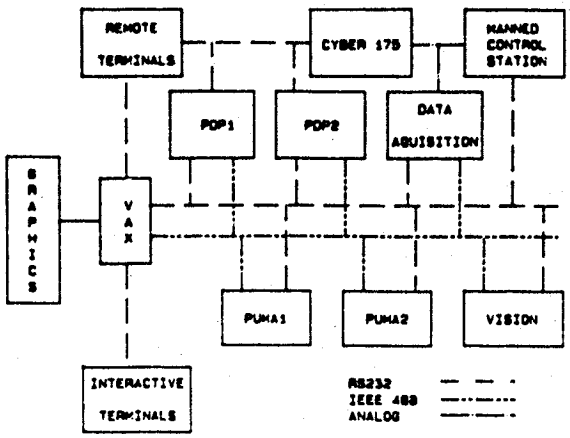


Figure 2. Telerobot system development cycle



Fiugre 3. ISRL configuration

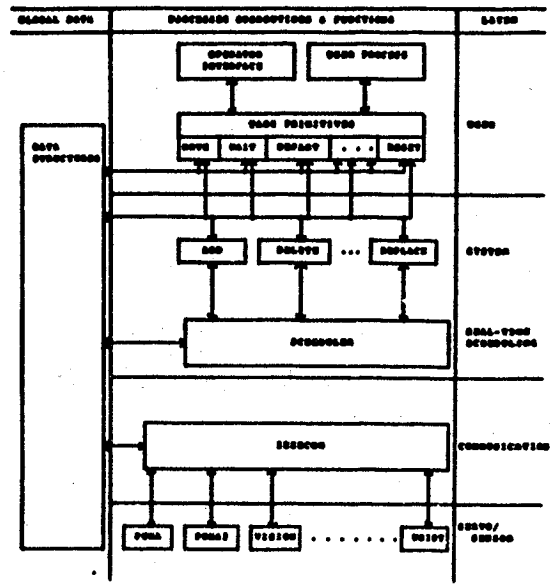


Figure 4. TRSS block diagram

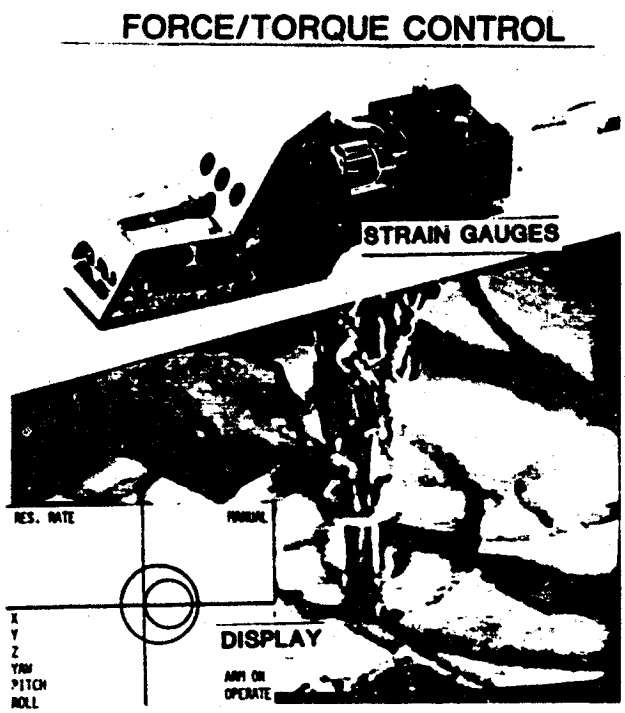


Figure 5. Force/torque sensor display

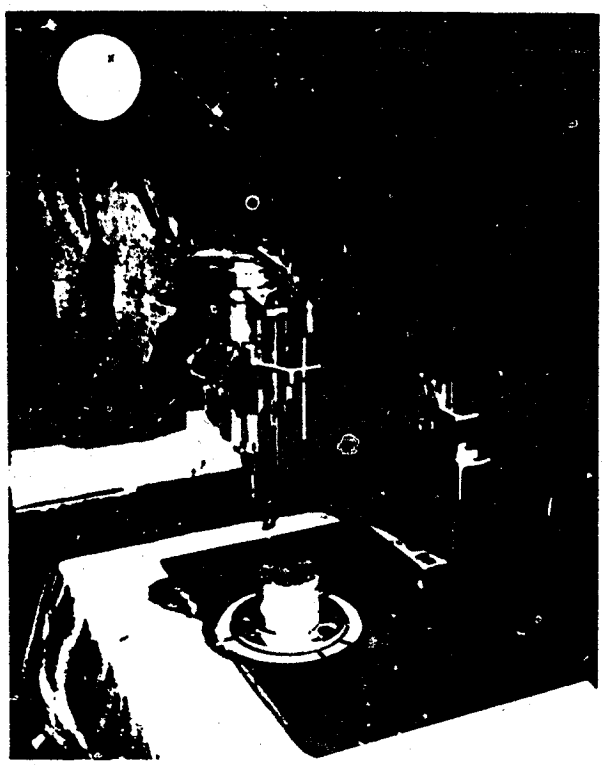


Figure 6. Simulated servicing task