AIAA-94-1267-CP

Virtual Environments for Telerobotic Shared Control

Brian K. Christensen

Deneb Robotics
Auburn Hills, Michigan

ABSTRACT

Traditional teleoperation depends solely on operator skill for effective task completion. This dependency limits the class of operations which were suitable for teleoperation. In many work situations today, external and operational requirements necessitate a more efficient operational scenario. The potential environments are more complex and dangerous and require integrated safety systems. One way to enable more effective control is to share control between the human operator and an intelligent computing system. This paper discusses the requirements and an implementation of a virtual environment for telerobotic shared control.

Efficient, effective and safe system operation depends on the operator's ability to make wise decisions. In order to make wise decisions the operator needs a clear understanding of the operational requirements, and an understanding of the external factors which affect the operation requirements. A virtual environment built upon a shared control interface can provide much of the information required for safe operation. The virtual world allows the operator to clearly visualize the task and provides feedback that is not otherwise available. The shared control interface verifies operational demands against system constraints. The virtual environment provides for a practice and training environment where mistakes can be made. Only after an operation has been cleared for execution does physical motion take place. Proper linking of human input with computer controlled heuristics greatly increases the safety level of the system.

The knowledge base that the computing system uses to perform decision making is called the *World Model*. Visual information from the *World Model* is displayed graphically as rendered, shaded, texture mapped animated polygons. The core of the shared control system is the computational engine that maintains, displays, visualizes and controls interaction with the *World Model*. The computational engine can be interrogated at several different levels. During operation: multilevel interaction defined by the program (mouse, keyboard, multi-modal input devices). The visualization system offers a high interactive operating environment, enabling users to change objects, display attributes, and elements of the *World Model* in real time.

High speed and optimal interaction with external input and teaching devices are required for effective interaction. Traditional teleoperation allows the user to effect immediate changes in manipulator configurations via an input device(s). Anything from a simple joystick to a force reflecting master have been utilized. The virtual environment uses these input tools and more advanced tools such as datagloves to direct user input into the graphics environment. This enables the use of the graphics environment for training, simulation, design interactions and will allow the *World Model* to interact and interpret operator commands.

2. INTRODUCTION

The integration of a graphical-based programming and control environment has the potential to enhance use and utility of robotic systems. Traditional off-line robotics programming methods address some of the needs for increased productivity of robotic systems. A more complete solution will address simplifying the programming process, direct linking of the graphics and control environment, the needs of cell calibration and model registration, and the needs of sensor- directed robot control. The graphic model part of the solution allows for extensive planning and conceptualization to occur without the use or need for actual equipment. The direct control side of the system allows for the seamless communication of commands and information from the graphical model to the device controller, and from the device controller and system sensors back to the graphics model. The sensor-based control requirements and the inclusion of calibration methodologies allow for confidence that the modeled world corresponds to the actual environment. The research described here represents our efforts to design and construct complete tools for enhancing robot or intelligent device planning, programming, and control processing.

3. HAZARDOUS ENVIRONMENTS: A TECHNOLOGY DEVELOPMENT FOCUS

An application focus is an important element of technology development. An application focus not only provides an understanding of which technology areas should be addressed first, but frequently suggests which technology approaches will be the most fruitful. For example, the highly structured nature of most manufacturing environments fostered the development of pick-and-place robots and strongly suggests that fixtures, not real-time sensing, is the most appropriate way to locate objects in the robot's work space. Teaching, in which the operator manually moves the robot to a location in the environment and stores that location (usually based upon encoder readings) in the robot's computer memory is the most common form of robot programming in manufacturing environments. During operation, the robot is instructed to repetitively return to some predetermined sequence of these previously taught locations. Fixtures assure that the workpiece is where it is expected and the robot carries out its action blindly. Production rates, not safety, are of primary concern. Personnel safety is typically provided by excluding people from the robot's workspace. If a part's location is not correct and an accident were to occur, the part or the robot might be damaged, but extensive damage is not expected.

Hazardous environments, however, place a premium on accident-free operation due to potential serious consequences of damage to the workpiece or environment. Rather than assuming that all objects in the robot's workspace are where they are expected, as in manufacturing environments, objects are always anticipated to be in unexpected locations in hazardous environments and all robot motions must be continuously monitored and validated.

Much of the original work leading to the development of graphical programming technologies at Sandia National Laboratories was performed to enable the application of robotic systems to hazardous nuclear and toxic environments. Due to the desire to limit personnel access, it is desirable to program robot systems within the context of a model of the environment, rather than use the teaching approach which would require operator entry into the hazardous environment. Without operator entry for close-up observation, teaching-based manual programming is very difficult. Thus, it is desirable to reduce the need for detailed operator programming and to transfer much of the reasoning task to the computer system (including sensors) controlling the robot. The knowledge base which the computing system uses to perform decision making is termed the *World Model*.

To allow the robot's control computer to intelligently reason about the environment, environmental sensing must be provided. Environmental sensing allows dynamic updating and validation of the *World Model* so the computer system's reasoning process is based upon a valid model. Real-time sensing increases operational safety by allowing the computing system to adapt the robot's movements to compensate for uncertainties in the *World Model*. Under no circumstances may the robot be allowed to increase the hazard associated with the already hazardous environment. If this occurred, robots would not be used because safety is an overriding issue when dealing with hazardous environments.

4. DEFINITION OF INTELLIGENT ROBOTS

Since the most common commercial robot applications are for repetitive operations requiring no sensing or decision making, robots are frequently envisioned as devices capable of only this type of behavior. Based upon the discussion above, many hazardous environments clearly require intelligent robot systems with more advanced operational characteristics. Thus, it is worthwhile to define what the term intelligent robot system means in the context of this paper.

Intelligent systems are systems which can make appropriate decisions when presented with situations. Such decisions include selecting motions which accomplish a goal without collisions with objects in the environment. John Hopcroft viewed robotics as the study of representing and reasoning about physical objects in a computer. Incorporating the more traditional mechanistic view of robotics provides the definition used here:

"Robotics is the integration of the sciences of sensing, representing and reasoning about physical objects in a computer coupled with electromechanical systems to carry out purposeful actions."[1]

In addition, intelligent robots possess skills with well characterized capabilities which can be used to accomplish tasks. A skill, for example, might include applying well-formulated knowledge about forces of interaction to perform in-contact operations, such as assembly, without operator intervention.

A reasonable goal for an intelligent robotic system is to serve as a supervised electromechanical system possessing sufficient intellect to serve in the place of a similarly supervised human. In the event a robot manipulator cannot accomplish its task due to errors, it either re-plans or

requests help from the supervisor, much as the replaced human would do. While intelligent robotic systems possessing the full range of capabilities implied by this goal may be beyond the current state of the technology, significant strides are being made.

5. A CONTROL ARCHITECTURE FOR INTELLIGENT SYSTEMS

In order to satisfy the performance characteristics described above, a structured computing system has been developed which allows incorporation of the wide range of capabilities required for the successful operation of robot systems in hazardous environments. The required capabilities range from fast, servo-level response based on sensor inputs (*e.g.*, follow a surface contour based upon interaction forces) to slower responses in which evaluation of various alternatives is involved (*e.g.*, planning a trajectory for the manipulator end point). The computing environment must integrate all levels of control into an efficiently executed control strategy which smoothly transitions from one control mode to the next. The basic computer architecture used for intelligent robot control is shown in Fig. 1. The hierarchical control environment has two main systems; the reasoning system in which high level control processes are performed, and the real-time control system in which fast response control processes are carried out.

Within the reasoning system, the computer constructs an approximate *World Model* based upon knowledge about the environment (*e.g.*, a map), robot characteristics (*e.g.*, kinematics) and heuristics about objects in the environment and their behavior (*e.g.*, physical limitations of the robot). The reasoning system also displays various aspects of the *World Model* (*e.g.*, geometric models, x-y-z plots, parameter traces) for operator understanding and assistance in decision making. As indicated above, this *World Model* is modified by sensory information and provides the foundation for automatically generating plans (*e.g.*, a collision-free robot trajectory) which are translated into robot manipulator motion primitives. The control processes taking place within the reasoning system can be quite complex and may require considerable computing time.

Within the real-time control system, servo control of the robot is accomplished. Responsiveness of the control system is extremely important as it is responsible for executing the robot motions developed by the reasoning system or supplied by direct intervention by the operator. A slow real-time control system would introduce delays between the commanding of movement and the execution of that movement by the robot, leading to instabilities in the system. Sensors and the *World Model* are employed to monitor the execution of these motions and to automatically perturb the robot motions if necessary to provide safe operation while accomplishing the desired task. Situations requiring direct operator control vary from the teaching of robot locations to the recovery from errors with which the robot control system is unable to cope. Experience with master/slave manipulators suggests that even highly trained operators experience considerable difficulty and tedium when executing remote manipulations.[2] Tedium can result in unsafe operation. Thus, direct operator commands are also monitored by the intelligent control system for compliance with safe operating practices and accepted procedures.

Within the context of Fig. 2, the operator takes on the role of planner and develops task plans for the robot manipulator to execute. However, the computing and sensory systems maintain their role as developers of approximate *World Models* and real-time controller. Much in the manner that the real-time control system perturbs the robot commands of the supervisor, the real-time control system now perturbs the commands of the operator within the context of a *World Model* and sensory system. The real-time control system assists the operator in automatically avoiding obstacles and executing controlled interactions with the environment while the operator performs the high-level task and path planning. Such computer assisted approaches to manual operation have proven effective in providing responsive robot manipulator systems capable of safe flexible operation when manually operated. [3]

5.1 Geometric Modeling

The intelligence of an intelligent robotic system resides in the *World Model* and algorithms for accessing and using the knowledge contained therein. The representation of the *World Model* is critical and the subject of much study because the efficiency of information retrieval determines the usefulness of the knowledge. [4] Robots deal with physical objects and, as stated by Hopcroft, reasoning about physical objects is the key element to constructing intelligent robots. A robot's ability to interact with a physical object is, to a large extent, defined by geometries. Effective geometric reasoning is the key element in constructing practical *World Models*.

5.2 The Role of Visualization Models

The *World Model* as described here encompasses much more than a model of geometric objects. The *World Model* is the visualization conduit providing the operator with insights into the performance of the control system and

displaying the results of control decisions. In addition to the geometry of an object, the model includes the kinematic, dynamic and motion characteristics of all movable objects in the environment. The *World Model* contains the knowledge to understand and predict how a controllable device will move and provides insights into how to plan a trajectory or path and in some cases may include knowledge on how to carry out tasks or complex operations. The *World Model* functions as the operator's window to the control of the intelligent system. It acts to validate command operations, provide real-time feedback to the motion or changes in the system, and provides an ideal place for quality assessment and data logging of all operator commands and the results of those commands. The visualization of the *World Model* is the key to operator understanding of system operation.

5.3 Simulation vs. Control

Over the past few years, much work has centered on the simulation of robots. In many cases this was the first step in off-line programming of these devices.[5,6] For static environments this may be all that is necessary. However, many of the control tasks of today are not static situations. These environments are typified by the challenges of remediation of hazardous waste or the manufacturing environment of flexible manufacturing systems. In these operations the work environment is not always well characterized before work begins, and may in fact change as the system operations are executed. The dynamic nature of these environments require a close coupling of the simulation and control systems. As updated information about the work environment is made available, the simulation or planning system must be capable of responding to the updated information and perturbing the commanded operations in response to these changes. The updated system information may be made available from sensors or from changing production requirements for flexible manufacturing systems.

In addition, the close coupling of the simulation and control system allows for safe integration of operator commands into the control of the robotic systems. The visualization system of the *World Model* acts as a filter to the operator commands. The *World Model* tests and in some cases perturbs the command before it is implemented. This is critical for the safe operation of systems requiring operator intervention.

5.4 Graphical Programming and Control

The goals of the development of the graphical programming and control system are faster and safer system operation and enhanced operator programming. The very nature of a 3-D representation of a modeled system enhances user operation, user understanding and aids user visualization. Graphic representations make it easy for the operator to program the system with icon selection and interactive manipulation of the graphic images. 3-D representation also enables whole-body collision detection not available anywhere else. Through simulation of camera views a camera is virtually presented to the operator. These views serve as an enhancement to live camera views, which may be limited and incomplete, and can enhance operator understanding of the work environment.

Control decisions which are deemed safe within the context of the *World Model* are communicated directly to the intelligent devices. Any translation is done on line and as part of the communication process between the reasoning system and the real-time control system. The motions caused as a result of these commands are fed directly back into the visualization system for recording and operator understanding and as a check on the validity of the commanded motions. In addition, sensors present in the intelligent devices locally verify modeled geometry and communicate discrepancies to the *World Model*. The *World Model* is dynamically updated and geometries constructed or modified as necessary. The updated information is then available for all future control decisions.

6. Implementation and System Description

A commercially available graphical robot simulation environment IGRIP from Deneb Robotics, Inc., was combined with the Sandia developed Generic Intelligent System Controller (GISC).[7,8] IGRIP was operated on Silicon Graphics workstations. Thus, we have built upon rapidly improving graphical computers and the large base of computer graphics capabilities. As the state of the art in high speed graphics computers and computer graphics improves, these advanced capabilities can be directly integrated. A critical area of development in the graphical programming of intelligent robots is providing high speed and efficient communication between the graphics *World Model* and the intelligent sub-system control elements. This is essential to allow graphics models to send commands to other processes and to allow other processes to send commands, interrupts and positional information to the graphics *World Model* for model updating and display.

The Low Level Telerobotic Interface (LLTI) developed jointly with Deneb Robotics to allow communication at the binary level between processes was the first attempt to tightly couple external data to the animation display. This

high-speed interface is essential to operator understanding of changes in the actual robot environment. It is also essential for proper visual feedback of operator-directed changes in the graphic environment using external input devices.

GISC allows the operator to control both the simulation system and multiple disparate programmable devices from the same control environment and in the same programming language. This is enabled by the use of the Sandia developed Robot Independent Programming Environment (RIPE) and in the Robot Independent Programming Language (RIPL).[9] The use of RIPL allows the system designer to communicate to many different programmable devices in the same manner. At one level the use of RIPL allows the programmer to communicate and direct the robot from different manufacturers in the same language syntax and structure. At another level through combination of the *World Model* and RIPL language, the operator can program any robot by simply graphically causing the robot model end point to move in response to external inputs. The benefits of the graphics environment is that the operator immediately sees the results of his directions and the operator is warned of potential collisions between the robots and objects in the environment. In fact, motion plans which would potentially cause collision are not permitted to be communicated to the controller and executed.

GISC employs a distributed computing environment for supervisory and real-time control of the robotic system, as shown in Fig. 1. The computing environment consists of a Silicon Graphics Iris (SGI) workstation which runs IGRIP for *World Model* display and animation, and also runs associated data communication processes. A Sun computer was also interfaced to the control environment for display of various non-geometric aspects of the *World Model*. Interfaced to the SGI are multiple Motorola 68030 single-board microprocessors resident in VME backplanes, which provide real-time, sensor-based control. The RIPE interface software translates the high-level commands from the SGI into sequences of low-level commands understood by the controllers and sensors. The robot controller and sensors are interfaced to the computing environment by the Sandia developed Intelligent Robot Operating Environment (IROE).[10] IROE is a real-time, multitasking operating environment built upon the VxWorks operating system (Wind River Systems, Inc.), and was specifically developed for the communication demands of real-time sensor directed-control of intelligent robot systems.

The graphics environment is an ideal environment in which to off-line program the robotic system. To minimize system downtime or to limit operator programming time on the system it is desirable to pre-plan the intended motions of the robotics system as much as possible. Accurate 3-D geometric models of the robot, its end effectors, and the fixtures and workpieces in the environment allows the robot programmer to carefully plan robot trajectories. Since the robot motions are simulated using accurate kinematic models, the pre-planned motions faithfully reproduce the actual motions of the robot. In addition, task execution time can be determined and analyzed for system bottlenecks.

Each robot typically has its own high-level programming language. The code to program the robot is written much like any other code, with the exception that the positions the robot is to move to are manually taught to the system. The integration of the graphical interface with an input device allows programming of the motions of the robotic system without having to write code. This can greatly speed the programming process and decrease the programming skill level required of the operator. Programming is thus realized by the operator causing motions of the simulated robot, the computer system remembering the commanded motions, replaying and displaying the commanded motions to the operator, and then with operator acquiescence, downloading the programmed motions to the robot controller and causing the motions to be executed.

The system software is written in such a way that any input device (*e.g.*, teach pendant, spaceball, force-reflecting master) can be used. The control commands from the input device are sent directly to the graphical system. The control system determines how these commands are to be interpreted, (*e.g.*, the movement of individual joints, or end point control based on a tool frame). The result of these commands are displayed real-time in the graphics environment. The motion information is combined with the collision detection capability contained in the *World Model*. Thus the operator can be alerted to the close approach of the robot to known objects in the environment.

The very nature of the graphics model allows for the ability to detect collisions between modeled objects. This ability is very important for the safe programming of robotics systems. It is undesirable for a robot to have an uncontrolled collision with objects in its environment. Such interactions are typically conducted at slow speeds, with force sensors measuring and controlling the forces of interaction. While an operator's attention is generally focused on the end of the robot and on the task at hand, another part of the robot, such as the elbow, may collide with objects in the environment. The collision detection

capability of the graphics environment detects and warns of all such impending collisions. In addition, both the parts selected for collision detection and the warning distance are user- selectable. This ability allows the task planner to select the objects of importance and to dynamically alter the warning levels based on the task at hand. This also allows for different collision detection capabilities depending on the skill of the operator.

6.1 System Operations

We have presently employed this system in two laboratory-scale systems and one full-scale demonstration for critical feature testing. In each case, the graphical control environment is used as the programming and control interface for GISC to produce a faster, safer, and more economical system. The first application of this technology was a laboratory demonstration relating to the remediation of nuclear waste buried in underground storage tanks.[11] The second application of this technology was the safe tele-operated inspection of a nuclear waste transportation cask.[12] The full- scale demonstration involved three robots working together to map a mock underground storage tank and then remove the simulated hazardous material.[13]

The graphical control environment described has been implemented and demonstrated in a Sandia laboratory test facility designed to demonstrate the robotic characterization and remediation of hazardous waste stored in underground tanks. The foundation *World Model* contains 3-D geometric models of the system generated from construction drawings. The system and its graphical representation can be seen in Fig. 3. The test work environment consisted of a gantry robot (Cimcorp XR6100) and a 2.4 by 2.1m rectangular tank filled to a depth of 0.6m with moist sand to represent the waste. The test tank contains both buried objects and pipes, as well as structures protruding above the surface of the waste to represent the types of obstacles that would be encountered during characterization and clean-up of actual storage tanks. The waste surface and some obstacles were deliberately not modeled to test the control system's ability to detect and map unknown obstacles. The *World Model* also contained models of the robot's kinematics and motion limitations and heuristics defining tasks such as mapping and waste removal. The initial robot motions were defined by operator, tested for safety and potential collisions within the *World Model* and executed. As motion occurred the robot model was driven by information received from the robot's encoders. As the sensor systems detected new information about the environment, such as waste surface profiles the data was communicated to the *World Model* and graphically displayed.

The second application of the graphics control environment is the inspection of nuclear waste shipping containers. This system also made use of the Cimcorp gantry robot in conjunction with a quarter-scale mockup of a waste transportation storage cask. Through use of the graphical interface, the operator can direct inspection tasks around the container. The operator can pick up tools and direct inspection tasks with the assurance that collisions will be prevented by the use of the graphics control environment. This capability will be essential for safety inspections and during emergency operations. The geometric *World Model* and the graphics interface to the operator are used to graphically program the robot system and to verify safe operation. For example, prior to the start of a series of programmed operations, the computer compares the desired robot trajectories with the geometric knowledge contained in the *World Model*. Any unsafe trajectory (*e.g.*, a collision between the robot and a model feature) is detected and reported to the operator via the graphics interface prior to robot motion. The operator can then modify the proposed robot trajectory or program the robot by simply manipulating the robot image in the graphic interface. These modifications to the robot trajectories are then verified for safe operation by the *World Model*. The computer system automatically reprograms the robot's movements to include the operator's commands. Only collision-free robot trajectories are transmitted to the robot.

The full-scale demonstration encompassed and enhanced the capabilities developed in the laboratory demonstrations. The expanded capabilities included interaction and control of three different robot systems, increasing the system's ability to handle a variety of work environments, and a number of different types of tools and sensor types. The system consisted of: a RedZone RTI robot used for tank wall and weld inspection; a Schilling Titan 7F robot used for fine manipulation tasks mounted on the end of a 28-foot-long Spar hydraulic arm used for gross positioning. A drawing of the system can be seen in Fig. 4. Also the graphical representation can be seen in Fig. 5. The tooling included an ultrasonic mapping tool, a hydraulic cutter, and a small pneumatic chipper.

The increased number of robots and tools tested the system's ability to program and communicate to different systems in the same language and using the same programming tools. The control structure and the command flow from the reasoning system to the real-time controller allowed seamless communication and control to each device. It also demonstrated the ability to perturb

commanded motion from updated real-time information. As in the laboratory demonstrations, surface information was mapped by the sensors and included in the *World Model*. Based upon this mapped information, ideal task trajectories are generated. When these trajectories are executed the sensor system perturbs the motion to maintain a desired height above the surface, for example. The results of the perturbed motions are fed back into the graphics system informing the operator of the changes made to the original desired trajectory.

In addition, the use of virtual camera views for operator feedback was implemented. The operator's view of the graphics representation of the *World Model* was adjustable. Through use of a set of dials the operator could modify the scale, translation and rotation of the view or views as he/she desired. This ability allowed for more detailed inspection of those areas of the model which were of particular interest. It should be noted that full collision detection of the entire model environment was always available regardless of what was displayed on the screen. In the particular case of removing and replacing tools from the tool rack, the virtual camera view was clearer and more valuable than the actual camera view which was cluttered and often obscured the desired information.

7. RESULTS & CONCLUSIONS

The use of a graphical control environment for robotic systems can accelerate tasks such as the removal of waste stored in underground storage tanks. Advanced 3-D geometric modeling concepts can allow robot motion planning and thus, facilitate programming the system without manual code generation. This greatly reduces the requirements for detailed, step-by-step programming by skilled robot programmers. The geometric model can interpret operator manipulations of the graphical representation to automatically program the robot to respond in the manner desired by the operator. In addition, the *World Model* can also be used to validate operator commands to the robot system to ensure safe operation during manual control. Graphical display of the results of the operator's robot commands can provide operators with perspectives not available from direct video viewing commonly used in the tele-operation of remote systems. This increases system safety by warning of impending collisions, and is especially important for impending collisions away from the robot end effector where the operator's attention is frequently focused. If an operator's command could result in a collision, the control system, with reference to the *World Model*, can prevent execution of the command by the robot and communicate the source of the problem to the operator through the graphics interface.

8. ACKNOWLEDGMENT

Much of the work described in this paper was performed while the author was at Sandia National Laboratories. The author would like to acknowledge the work of William Drotning, Peter Boissiere, William Davidson, Michael Griesmeyer, Sig Thunborg, and Ray Harrigan in developing the software and hardware interfaces that made Graphical Model based control possible.

9. REFERENCES

1. Hopcroft, J., (1986). Impact of Robotics on Computer Science. Communications of the ACM, Vol. 29, #6, 486-498.
2. Martin, H. and Kuban, D., (1985). Teleoperated Robotics in Hostile Environments, Dearborn: Robotics International of the Society of Mechanical Engineers.
3. Boissiere, P., and Harrigan, R., (1989). An Alternative Control Structure for Telerobotics. Proceedings of the 1989 NASA Conference on Space Tele-robotics, vol 5, p. 141.
4. Brooks, M., and Lillekjendlie, B., (1989). A Graphical Programming Environment for a Grinding Robot, Proceedings of 20th International Symposium on Industrial Robots, Tokyo, Japan.
5. Tarnoff, N., Jacoff, A., and Lumia, R., (1992). Graphical Simulation for Sensor Based Robot Programming, accepted for publication in Journal of Intelligent and Robot Systems
6. Tarnoff, N., and Lumia, R., (1988). The Role of Off-line Robot Programming in Hierarchical Control, Second International Symposium on Robotics and Manufacturing Research, Education and Application, ISRAM, Albuquerque, NM.
7. Deneb Robotics, (1992). IGRIP, Auburn Hills, MI.
8. Harrigan, R., (1992). Generic Intelligent System Controller Review, in Progress
9. Miller, D., and Lennox, C., (1990). An Object-Oriented Environment for Robot System Architectures, Proceedings of the 1990 IEEE International Conference on Robotics and Automation, vol. 1, p. 352.
10. Davidson, W., (1992). IROE, An Intelligent Robot Operating Environment, to be published.
11. Christensen, B., Drotning, W., and Thunborg, S., (1991). Model Based, Sensor Directed Remediation of Underground Storage Tanks, Proceedings of the 1991 IEEE International Conference on Robotics and Automation.
12. Drotning, W., and Bennett, P., (1992). Graphical Models for Simulation and Control of Robotics Systems for Waste Handling, to be presented at the 3rd International High-Level Radioactive Waste Management Conference
13. Christensen, B., Griebenow, B., and Burks, B., (1991). Graphic Model-Based Control of Robotic Systems for Waste Remediation, Transactions of the American Nuclear Society, 64, 609.

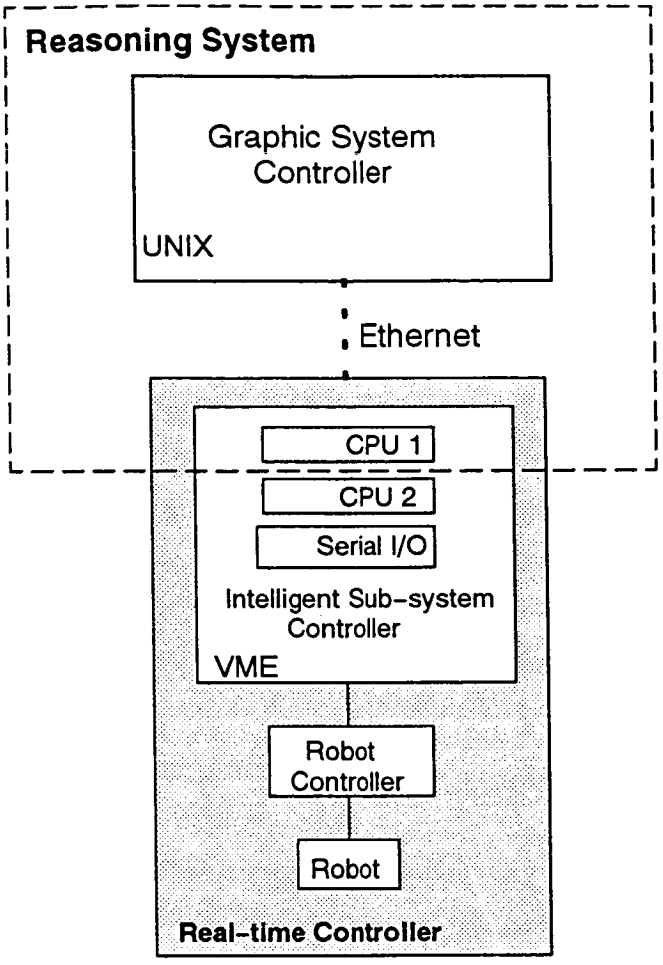

Fig. 1 System Architecture

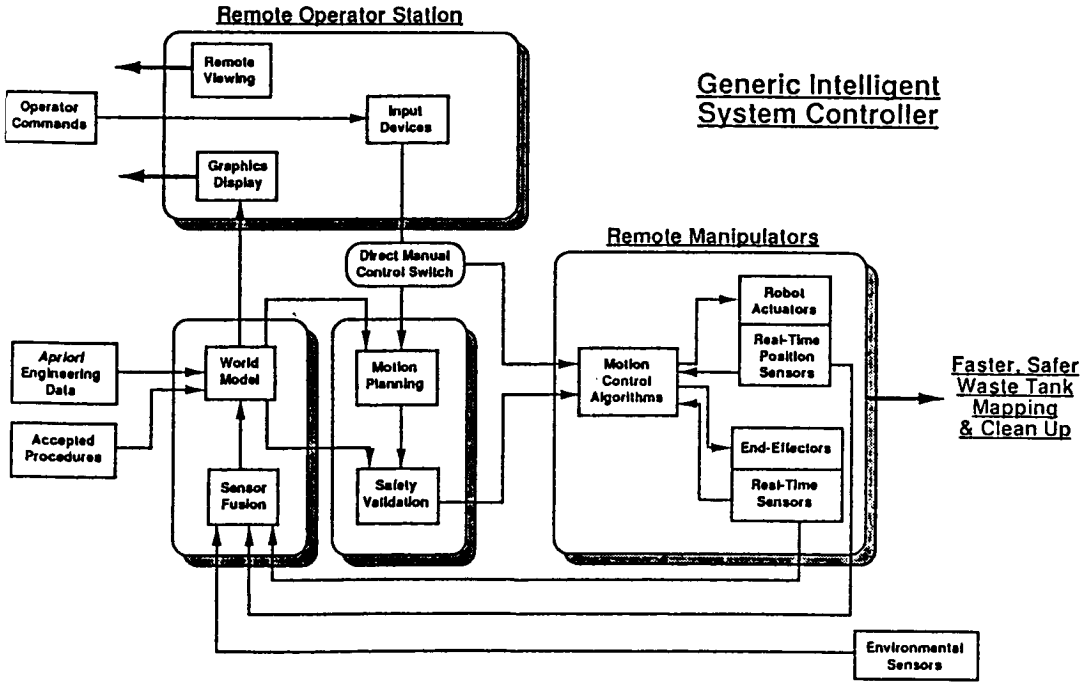

Fig. 2 Distributed GISC Environment

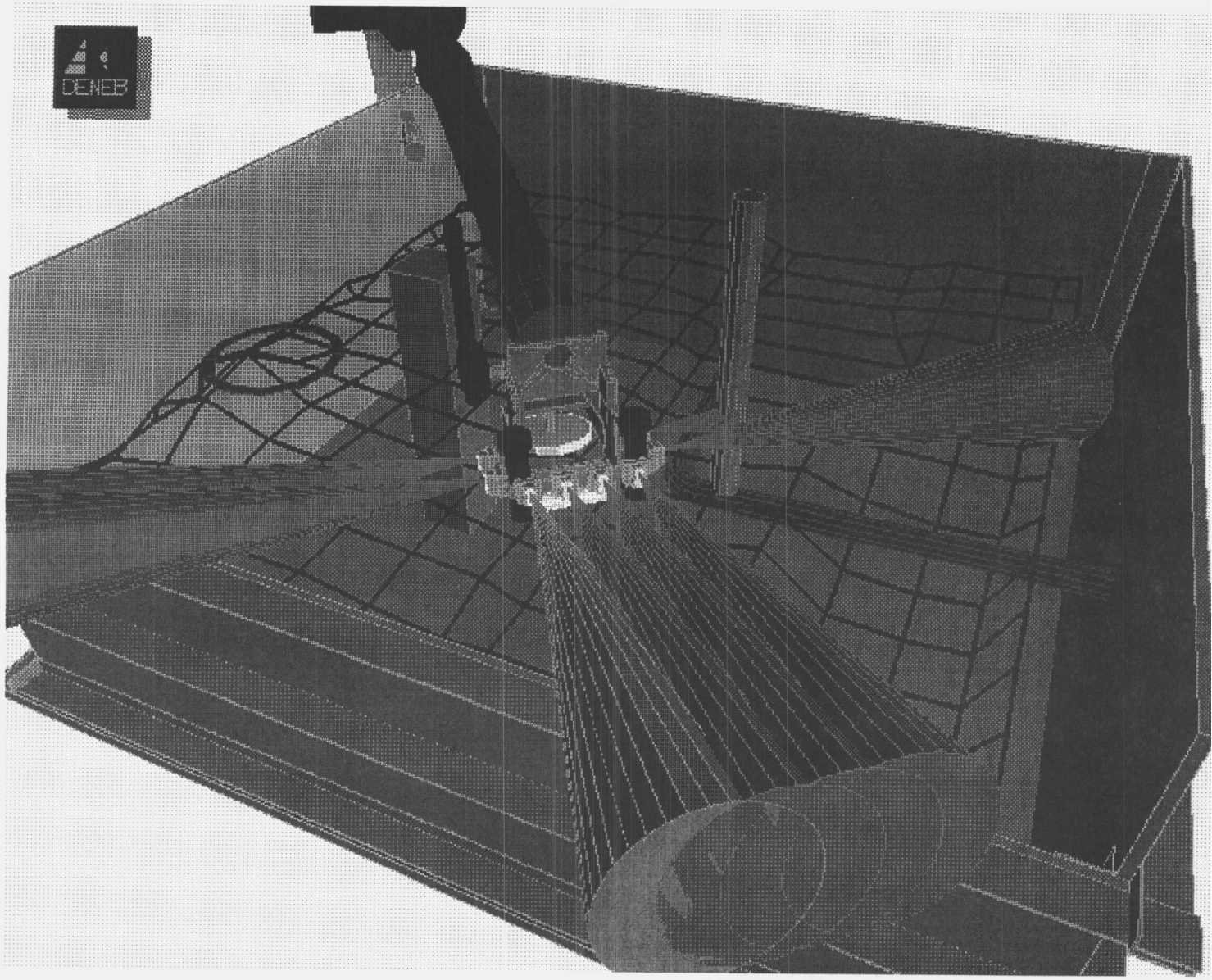

Fig. 3 Graphical Representation of Waste Remediation Test Bed

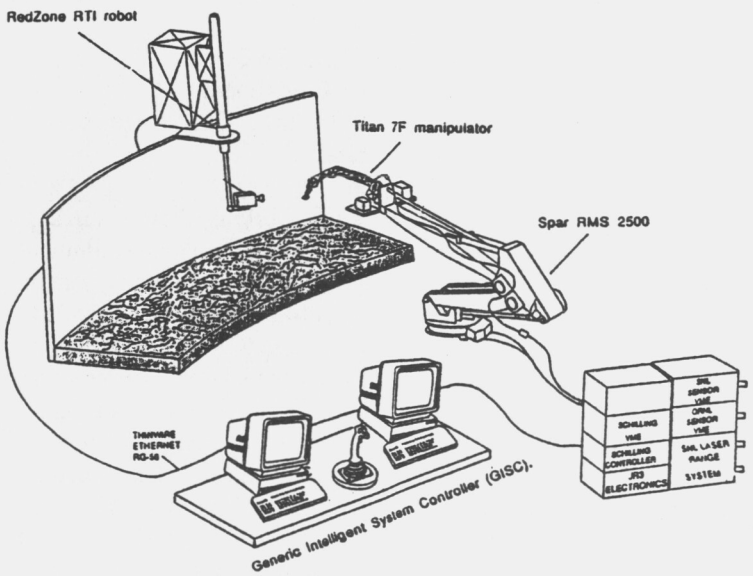

Fig. 4 Full-Scale Waste Remediation System

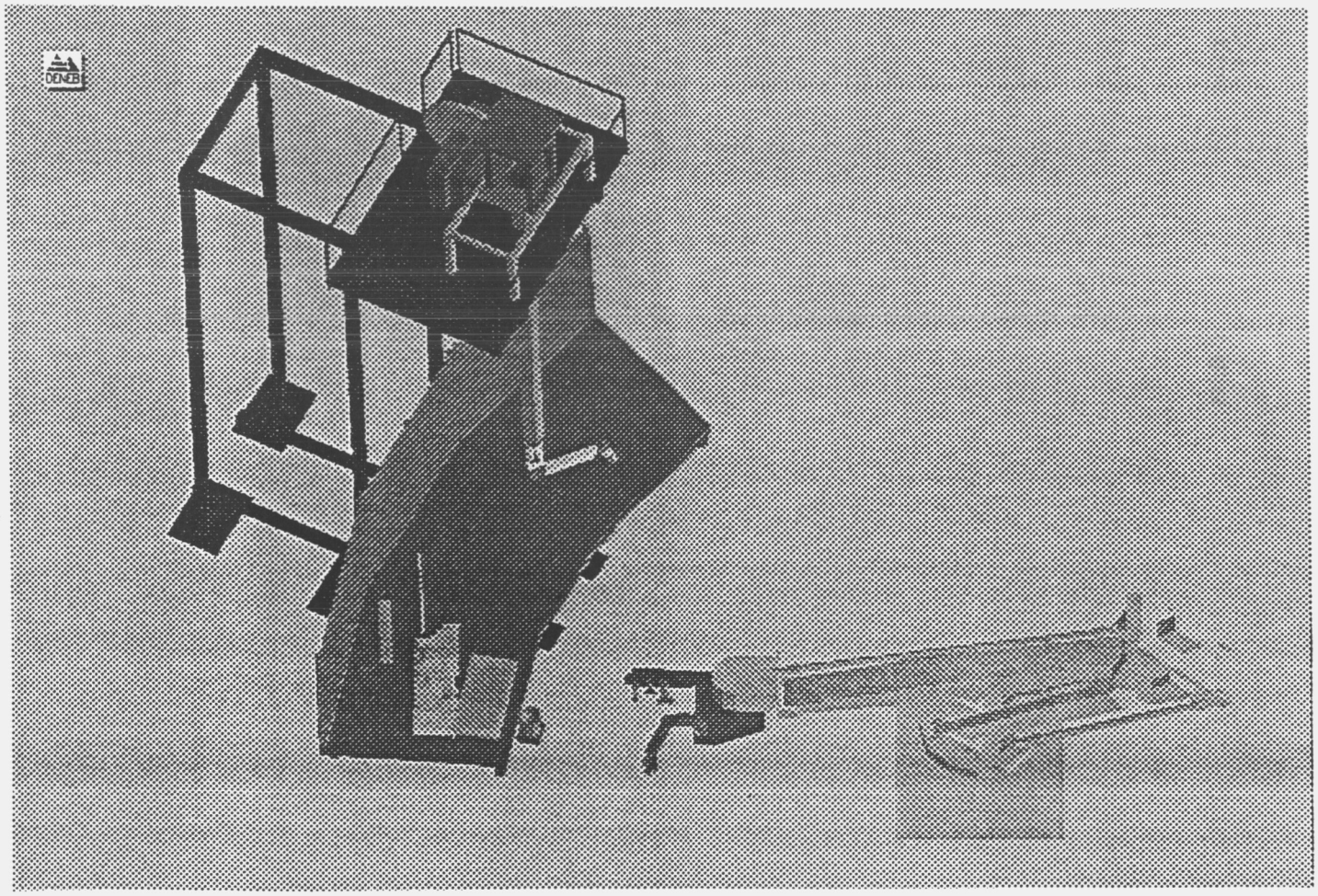

Fig . 5 Graphical Representation of Full-Scale Waste Tank Remediation System