N94-33666

GRAPHICAL PROGRAMMING: A SYSTEMS APPROACH FOR TELEROBOTIC SERVICING OF SPACE ASSETS

James T. Pinkerton
Department of Computer Science
University of New Mexico
Albuquerque, NM 87185

Michael J. McDonald
Robert D. Palmquist
Sandia National Laboratories
Intelligent Systems and Robotics Center
Albuquerque, NM 87185-5800

Richard Patten
Oceaneering Space Systems
1620 Texas Avenue, Suite C-9
Webster, TX 77598

ABSTRACT

Satellite servicing is in many ways analogous to subsea robotic servicing in the late 1970's. A cost effective, reliable, telerobotic capability had to be demonstrated before the oil companies invested money in deep water robot serviceable production facilities. In the same sense, aeronautic engineers will not design satellites for telerobotic servicing until such a quantifiable capability has been demonstrated.

New space servicing systems will be markedly different than existing space robot systems. Past space manipulator systems, including the Space Shuttle's robot arm, have used master/slave technologies with poor fidelity, slow operating speeds and most importantly, in-orbit human operators. In contrast, new systems will be capable of precision operations, conducted at higher rates of speed, and be commanded via ground-control communication links. Challenges presented by this environment include achieving a mandated level of robustness and dependability, radiation hardening, minimum weight and power consumption, and a system which accommodates the inherent communication delay between the ground station and the satellite. There is also a need for a user interface which is easy to use, ensures collision free motions, and is capable of adjusting to an unknown workcell (for repair operations the condition of the satellite may not be known in advance). This paper describes the novel technologies required to deliver such a capability.

INTRODUCTION

Graphical Programming uses 3-D animated graphics models as intuitive operator interfaces for the programming and control of complex robotic systems. This paper reviews several example robotic systems that use Graphical Programming as practical operational systems. The general approach to implementing Graphical Programming systems at SNL is then examined together with a description of the software environment used to implement general Graphical Programming concepts. Lessons learned from applying Graphical Programming to prototypical waste cleanup robotic system control are then reviewed with suggestions for new directions for future technology development.

The US Department of Energy Office of Technology Development (DOE OTD) has sponsored the Robotics Technology Development Program (RTDP). Development of innovative technologies for programming and controlling advanced robotic systems for application to the clean up of hazardous radioactive waste has been a focus of the RTDP. Of particular concern has been the development of generalized control approaches which automate clean up operations to reduce the time and cost of waste clean up while providing very high safety. Many of

*This work was performed at Sandia National Laboratories supported by the US Department of Energy under contract number DE-AC04-76DP00789.

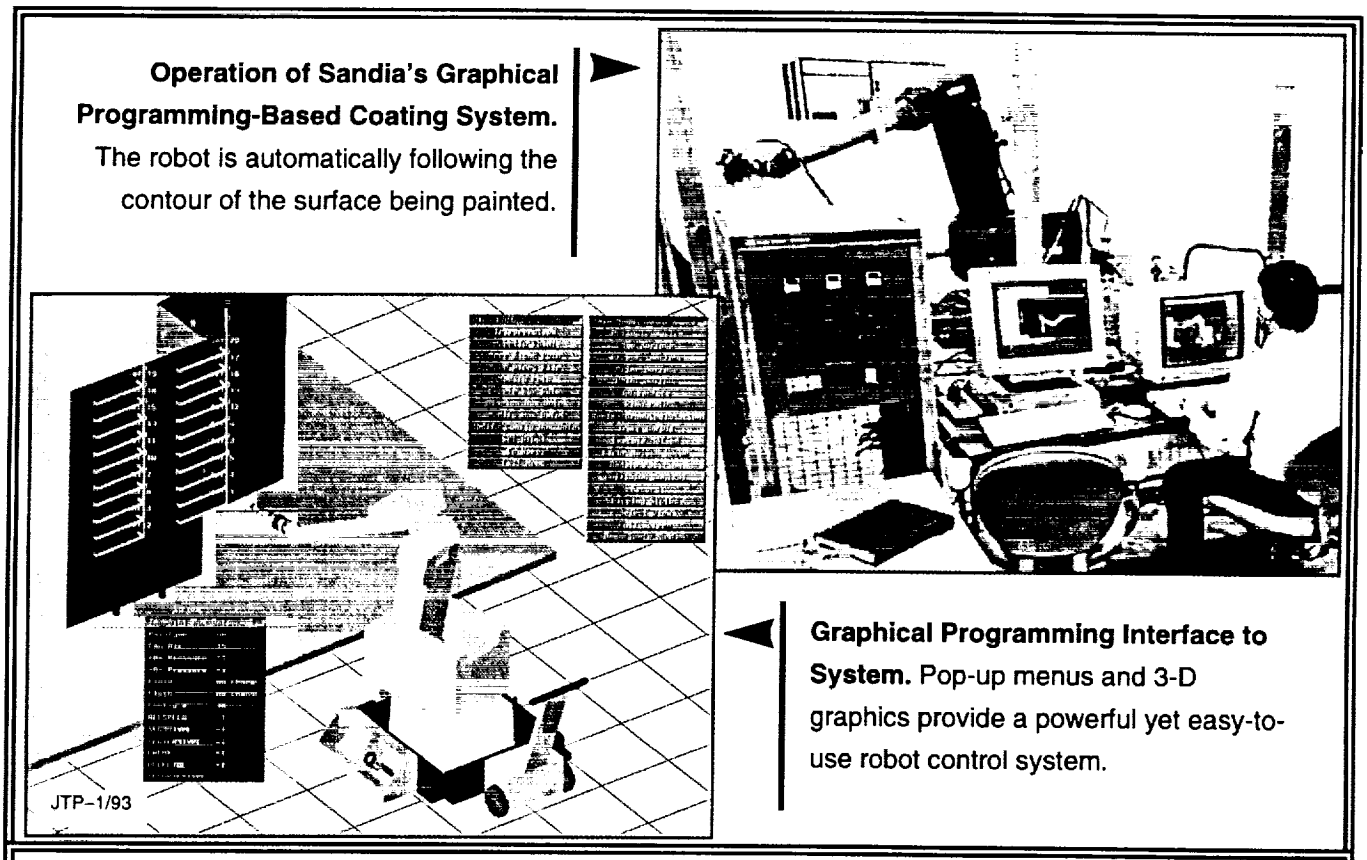

Figure 1: A Graphical Programming System for Applying Hazardous Coatings

the technologies developed in this program are directly applicable to the telerobotic servicing of space assets. The operational issues of space applications are in many ways similar to hazardous waste applications. For example, redundant safety, operator involvement, and robust operation are key issues in both environments. Therefore, technologies developed for hazardous waste environments, such as model based motion preview, modular sensor integration, and remote intelligence are appropriate for telerobotic operation in space. Model-based control approaches have proven to be very effective in allowing non-experts to easily program robot systems. This approach, coupled with animated graphics operator interfaces which employ advanced visualization software technologies both to communicate information to the operator and to facilitate operator communication to the robot system, reduces the robot operator training requirements while decreasing the programming time for even complex operations.

OVERVIEW

Figure 1 shows a prototype robot system with a Graphical Programming interface. The graphic representation of the robot allows an operator who is not an expert robot programmer to easily interact with the robot's supervisory system control software and command robot motions. The photograph shows the actual robot and its graphical model. In a typical prototype robot system, as shown in Figure 1, the robot's supervisory control software:

- Translates commanded tasks into graphical robot motions.
- Simulates and analyzes robot motions to check for safety.
- Commands the robot to execute motions that have been determined to be safe.
- Monitors the robot's motions to verify task compliance.
- Updates the graphics model as tasks are performed by the robot.

The Graphical Programming paradigm, as developed by Sandia for application to robot system control, broke new ground by integrating sophisticated 3-D graphics modeling technology into the real-time control of robot systems. The real-time updating of the graphics model to allow continual validation of robot motions distinguishes Graphical Programming from conventional off-line program-

ming. Off-line programming requires complete knowledge of the robot's workspace while Graphical Programming, with the key attributes of environmental sensing and dynamic model updating, allows operation with incomplete knowledge (see next section). Thus, conventional off-line programming is a tool to verify robot programs before execution while the graphics model of a Graphical Programming system is an integral part of the high-level system control environment.

The next section, *The Graphical Programming Approach*, describes Graphical Programming as a general approach to designing robot supervisor systems. The following section, *Graphical Programming System Examples*, describes several robot control systems that use Graphical Programming or significant concepts from Graphical Programming. The *Sandia's Graphical Programming Systems* section describes Sandia's particular approach to implementing Graphical Programming systems, several tools that Sandia uses in designing those systems, and important features that Sandia has implemented in various Graphical Programming systems. Finally, the paper concludes with *Future Work* and *Conclusions* sections to briefly describe Sandia's current plans and directions.

THE GRAPHICAL PROGRAMMING APPROACH

Graphical Programming systems use graphic-based, robot simulation systems in the operator interface for programming, controlling, and monitoring complex robot systems. The Graphical Programming Supervisor software module commands, controls, and monitors robots and sensors in the task, or high-level control loop while the robots and sensors use local controllers to control the low-level aspects of the robot including servoing and autonomous operations. The Supervisor monitors the sensors used for low-level tasks (including encoders and force sensors) and other sensors (including laser range finding sensors) to maintain the world model's accuracy.

Simulation and monitoring are integral functions of the Supervisor software. Robot tasks are simulated before they are performed and the simulation system's safety validation functions determine whether the task can be performed safely. System operators who have proper access control can override safety systems if they determine that the safety analyses are too conservative for a particular task. While the tasks are being performed, the Supervisor slaves the simulation system to the robot's motion sensors and monitors the robot to verify that the task was performed as simulated, or, as with sensor-controlled tasks, to track the real-world effects of the sensors. The Supervisor can also interrupt robot motions that excessively deviate from predicted motions or result in entry into hazardous regions. Force compliant motions are performed at the Subsystem level, making the system tolerant of long network delays between the Subsystem and Supervisor while still providing stable motion. This also minimizes data bandwidth requirements.

With the real-time tracking inherent in Graphical Programming, the Supervisor can command sensors to locate new or moved objects (i.e., fixtures and workpieces) in the environment and instantly display those sensed objects in the graphic environment. Engineers can also use these up-to-date models as an accurate base to design workcell modifications when requirements change. If the Supervisor is space-based, the effect of emergency stops and other unplanned events are immediately represented in the world model and can be quickly and effectively acted on by the system operator.

The real-time tracking also provides a continual quality audit function from development to retirement. In any development effort, the robot is commanded to move many times to test robot functions. By using Graphical Programming, the Supervisor simulates the robot before each motion and monitors the motions when they are performed. This cycle closes the loop on simulation and experimentation by allowing the system to verify its own simulation accuracy each time the robot moves. In effect, each robot motion is an experiment that verifies the Supervisor's safety systems. This lets the developer identify and eliminate simulation model inaccuracies in the early phases of system integration, and allows operators to verify the model throughout the life of the system.

Graphical Programming systems are largely data driven. Supervisors can be rapidly modified for new robot systems by modifying the system's world model. Tasks can be redefined within the model without changing the Supervisor program. Only code that reflects fundamental requirement changes needs to be written to extend a Supervisor for a new robot system. For example, a Graphical Programming Supervisor that was designed for remote retrieval of orbital replacement units could be rapidly modified to control a space-based maintenance operation by changing the geometric (graphic) and motion (kinematic) models and by modifying a few very task-specific command menus.

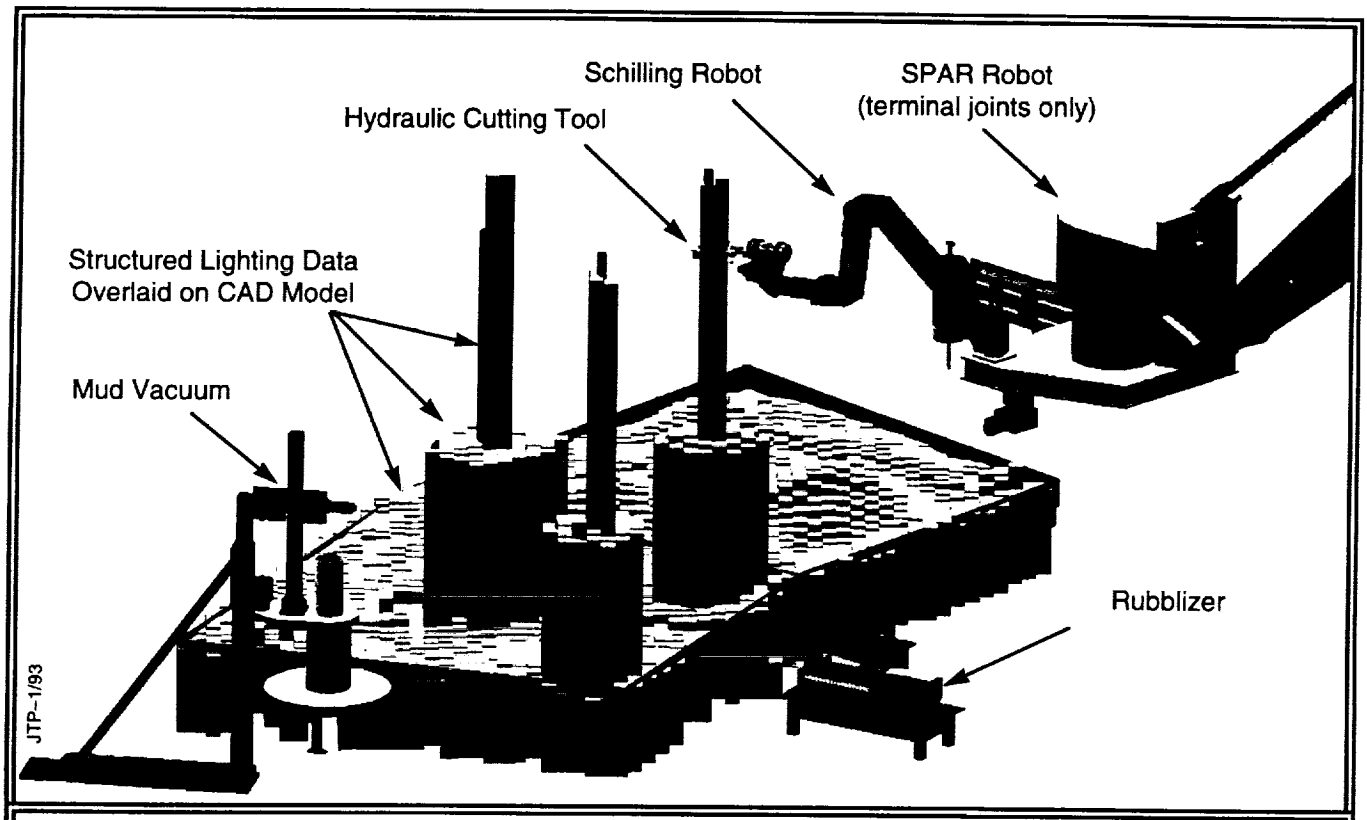

Figure 2: Graphic Model of Underground Storage Tank System at Hanford With Integration of Structured Lighting Data

Graphical Programming systems bring advanced technology to the robot operator. With Graphical Programming, the operator can visualize and understand the result of complex commands before moving any machinery. Advanced planning and sensor-based control systems are integrated into Supervisors without taking control away from the operator. The operator can see intended robot motions from any angle, position, or magnification and can modify the motions to accommodate for conditions that the automation and planning subsystems did not resolve. The operator can analyze motions by using standard simulation system analysis tools (including collision and near-miss detection) and optimize the motions by using sophisticated input devices (including spaceballs, dial boxes, and robot teach pendants).

Graphical Programming systems improve system safety over competing systems in several important ways.

- Hazards are predicted through simulation and locked out through program control.
- The operator is warned of motions that would cause near-collisions (with the near-miss distance set by the operator).
- Motions that could cause collisions cannot be commanded to the robot unless the operator has specific override permission.
- The quality audit function of linking simulation to monitoring is a thorough method for verifying that safety calculations are correct.
- The Supervisor world model is consistent with the real world and, therefore, safety checks remain accurate even when the robot's operating environment changes.
- Software reuse allows Supervisory software to be quality-verified in many situations.
- Advanced technologies can be integrated to improve operator efficiency without reducing safety.

GRAPHICAL PROGRAMMING SYSTEM EXAMPLES

In 1990, Sandia demonstrated that Graphical Programming provides a dramatic improvement in ease of operation and operational safety when compared to teleoperation [1, 2, 3]. The Graphical Programming Supervisory system used a real-time computer subsystem to control a gantry robot, several sensor systems (including structured lighting,

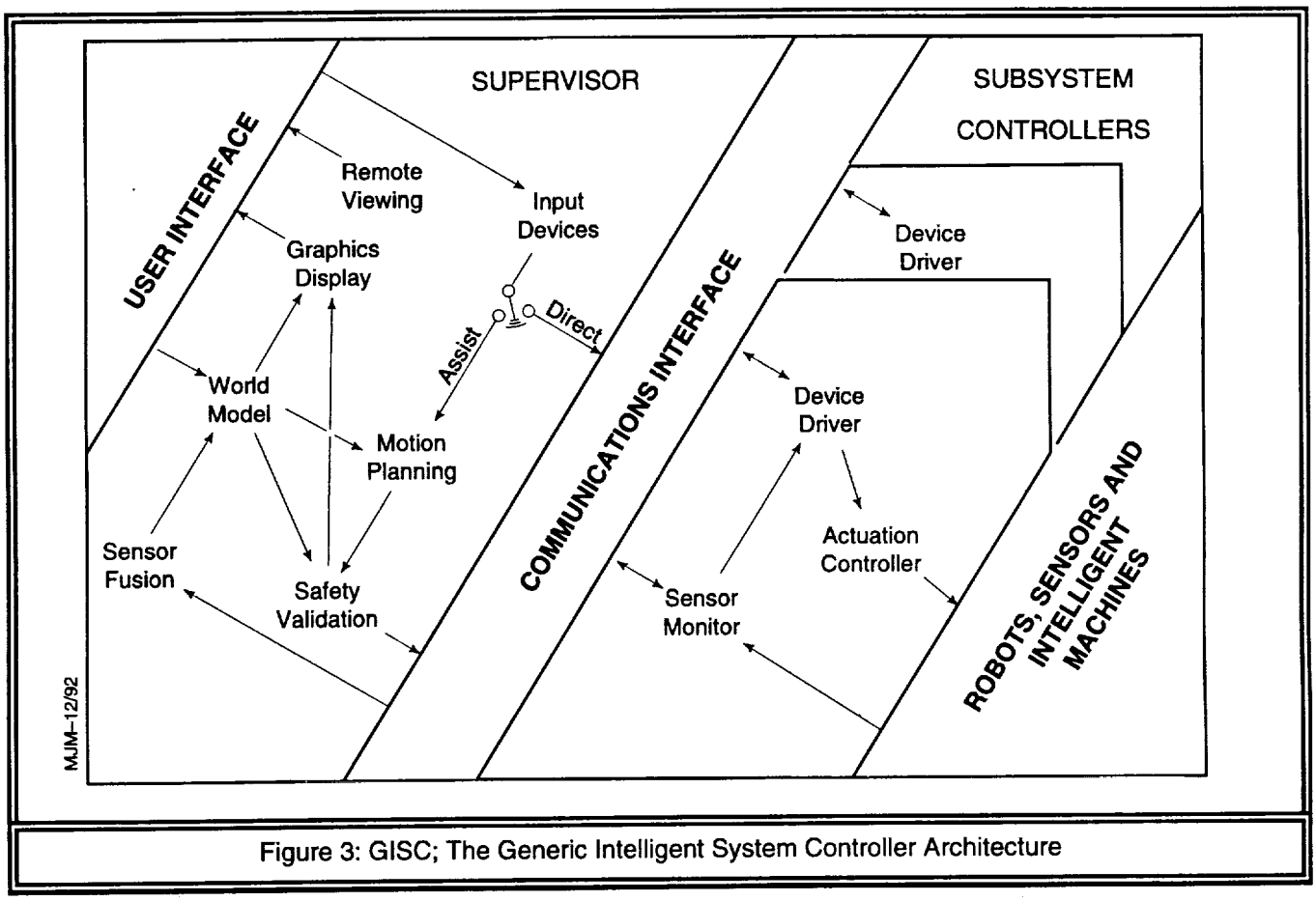

Figure 3: GISC; The Generic Intelligent System Controller Architecture

ultrasonic and magnetometer), and special tools. The Supervisor imported a contour model of a surface that had been measured with a structured lighting system into the simulation model of the robot workspace. Operators could graphically program robot motions in the workspace with menus and a spaceball input device. The Supervisor then used motion preview, collision detection, and joint travel checking routines to verify the safety of programmed motions before giving the operator an option to execute the motions on the robot. Visitors to the lab were trained in minutes to safely command and safely control the powerful gantry robot.

In 1991, the RTDP sponsored development of a multi-robot demonstration system for underground storage tank operations at the Hanford site near Richland, Washington [4]. The Supervisor here simultaneously controlled SPAR, Redzone, and Schilling robots, and monitored several sensor systems. The effort demonstrated that diverse intelligent subsystems, developed at different and distant laboratories, and each with unique control systems, could be rapidly and effectively integrated into a single system and controlled with a Graphical Programming-based Supervisor.

In 1992, Sandia developed four new Graphical Programming supervisory systems. These systems were a CIMCORP survey gantry robot, a GMF painting robot for applying hazardous coatings (Figure 1), a Schilling ESM long-reach painting robot, and an enhancement to the underground storage tank system developed in 1990 (Figure 2). In addition, Savannah River Technical Center (SRTC), in consultation with Sandia, implemented a Graphical Programming Supervisor for a gantry telerobot that had an added ability to take control away from an operator who commanded unsafe motions through the direct control master/slave input devices. All these diverse systems shared significant portions of their Supervisor application software and differed mainly by the unique tasks that each system needed to perform.

Sandia recently showed that a Graphical Programming Supervisor could control a robot Subsystem over the Internet using minimal bandwidth. The Supervisor was located at Hanford, Washington, and the gantry robot subsystem was located at Sandia in Albuquerque, New Mexico. The communications link included many hops and shared a 56 KB link with the rest of our group. The distance was transparent to the user because of local

previewing of operations and a clean division between the Supervisor and Subsystem.

Besides Sandia, other institutions are using significant concepts or techniques related to Graphical Programming. In 1990, the Jet Propulsion Laboratory (JPL) described a system that used computer graphics techniques to enable the human operator to both visualize and predict detailed 3-D trajectories of teleoperated manipulators in real time [5]. In 1991, MITRE Corp. reported on a virtual image concept that allowed software-based graphical monitoring to monitor teleoperation tasks in real time [6]. These efforts foreshadow the wide use of Graphical Programming as a telerobotic interface.

SANDIA'S GRAPHICAL PROGRAMMING SYSTEMS

Sandia constructs Graphical Programming systems in unique ways to facilitate rapid prototyping and to reduce development costs. The following points outline the major differences that the remainder of the section describes in greater detail:

- Sandia uses the Generic Intelligent System Control (GISC) approach, an RTDP approved method for robot system integration.
- Our Supervisors use high-performance, Unix-based graphic workstations.
- We use dedicated real-time computers for high speed and low-level robot control.
- We link the real-time computers to the Supervisor's computer with standard communication interfaces.
- We use a Sandia-developed generic communications message protocol to command robot motions.
- We rely on our extensive library for robot system development.
- We use commercial three-dimensional simulation and visualization systems in our Supervisors.

Sandia develops Graphical Programming systems by using the GISC approach [7, 8, 9, 10, 11]. GISC is a general approach to constructing robot systems that was developed by the RTDP. Figure 3 is a diagram of the GISC approach to designing complex robot systems. Sandia's Graphical Programming Supervisors are examples of Supervisors (Figure 3) designed using the GISC approach. GISC Subsystem Controllers (Figure 3) control the low-level aspects of the robots and sensors through Device Drivers. This low-level control includes servoing, direct teleoperation, and autonomous task execution.

The GISC system (Figure 3) starts with a World Model that is generated by the user from *a priori* engineering data. As shown in Figure 3, the GISC Supervisor:

- Interprets user commands made with menus and other Input Devices into tasks that are planned with the Motion Planner.
- Tests tasks for safety with the Safety Validation module and displays test results through a Graphic Display.
- Commands the robot subsystems to perform tasks that are verified as safe.
- Updates the World Model from sensor data generated by the subsystems.

Sandia's approach to Graphical Programming uses GISC to define the overall control algorithm and to set feature requirements for the supervisors and subsystems. Our Graphical Programming Supervisors perform all the functions of the GISC supervisor and interface with GISC Device Drivers. In this way, our Supervisors are plug-compatible with other GISC Supervisors and can be developed in parallel to Supervisors that do not need the advanced features of Graphical Programming.

Supervisors and robot Device Drivers are separate programs and normally run on different computers. Supervisors communicate with the Device Drivers by using conventional communication media including TCP/IP and RS232. These standard communication systems are enhanced for the application programmer with Sandia's Intelligent System Operating Environment (ISOE) and GENeralized Interface for Supervisor and Subsystems (GENISAS) [10] communications libraries. The ISOE library multiplexes synchronous and asynchronous commands, status queries, and data exchanges over a single synchronous communications link. The GENISAS library links ISOE to user functions and data from application programmer defined tables of functions and data. Together, these libraries free the application programmer from writing code that links commands and queries made through the communications link to robot control functions and data transfers.

We design our subsystems to respond to generically-defined commands that are broadly applicable to robot control. These generic commands are based on the Robot Independent Programming Environment and Language (RIPE/RIPL) developed

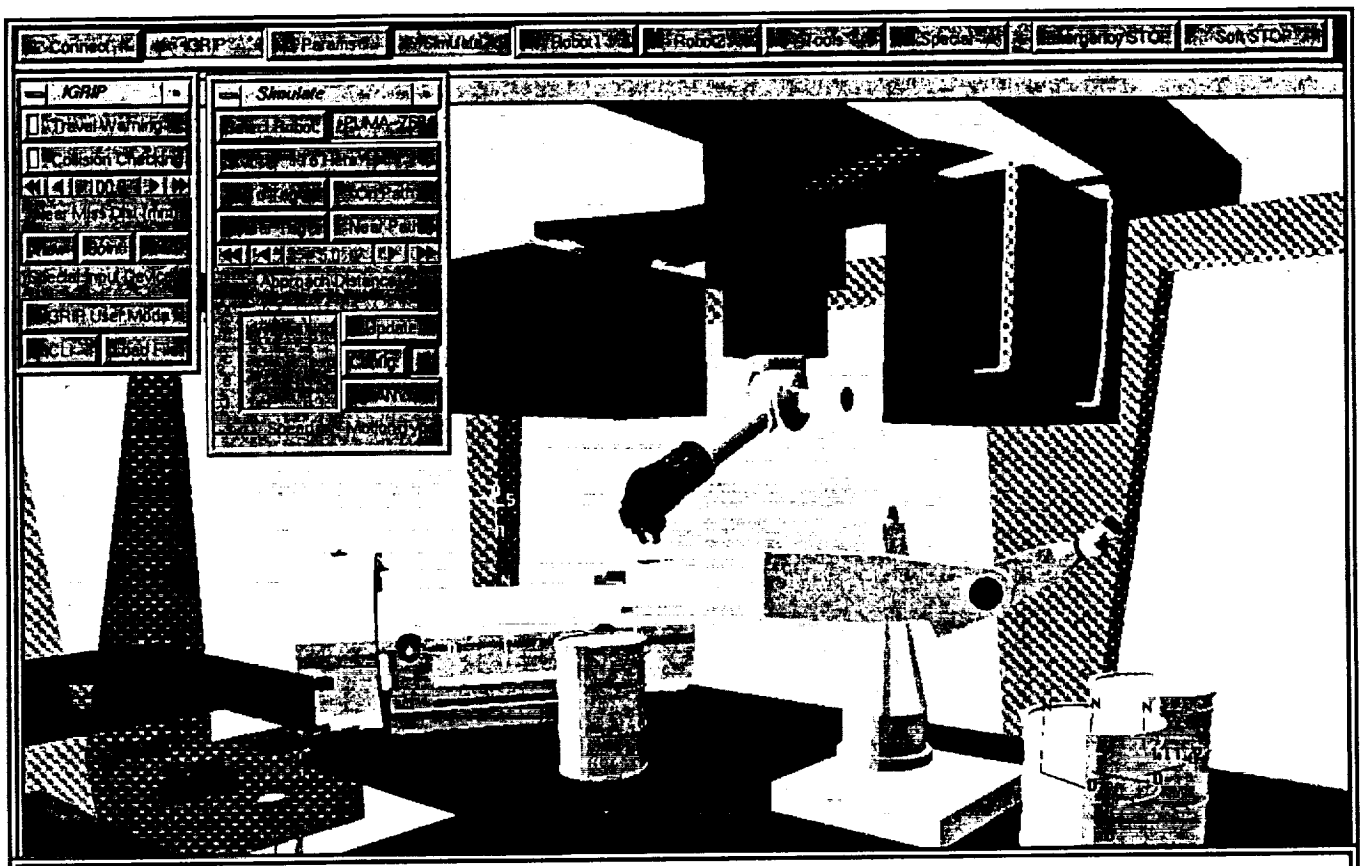

Figure 4: A Prototypical Graphical Programming System Written with the External Control Feature of IGRIP

at Sandia for autonomous systems [12]. Commands range from point-to-point and path moves to force-controlled and other sensor-based motions. Our GISC-designed Device Drivers that are written with RIPE/RIPL take full advantage of the GENISAS and ISOE libraries.

The generic command set and standardized communications libraries are the enabling technologies that allow the Supervisor programs to be data driven. Because dissimilar robots respond to the same command sets, the Supervisor only needs to be programmed to generate one command set to communicate with a wide variety of robots. In this way, only the portion of the World Model that describes how the robots will behave when given these generic commands and the portion that contains the geometry of the specific objects in the workspaces need to be changed to allow a Supervisor designed for one system to control a new and different system.

Our Supervisors use commercially-supplied robot simulation systems to simulate and graphically display robot motions and to perform routine analysis checks. This approach to developing Supervisors leverages from the commercially available simulation systems and frees Sandia from developing simulation and visualization subsystems. The approach also makes the Graphical Programming technologies easier to transfer to industry as the bulk of the system programs are already in the commercial arena. Finally, Sandia's product surveys indicate that any of several simulation systems could be used for Graphical Programming systems. This means that while our systems might be developed with one simulation system, the final application can use a different system.

Until recently, the Sandia-written application program for the Supervisors was contained inside the simulation system by using application programming languages. For example, the menus and high-level functions for the painting robot shown in Figure 1 were written with Deneb Inc.'s IGRIP [13, 14] using its Graphical Simulation Language (GSL) application programming language. While developing the application programs inside the simulation environment allowed Sandia to rapidly develop its initial Graphical Programming Supervisors and test various Supervisor system algorithms, ultimately, it limited the scope of the Supervisory program to only include functionality supported by GSL. For example, a translator was required to access ISOE- and GENISAS-driven subsystems. This

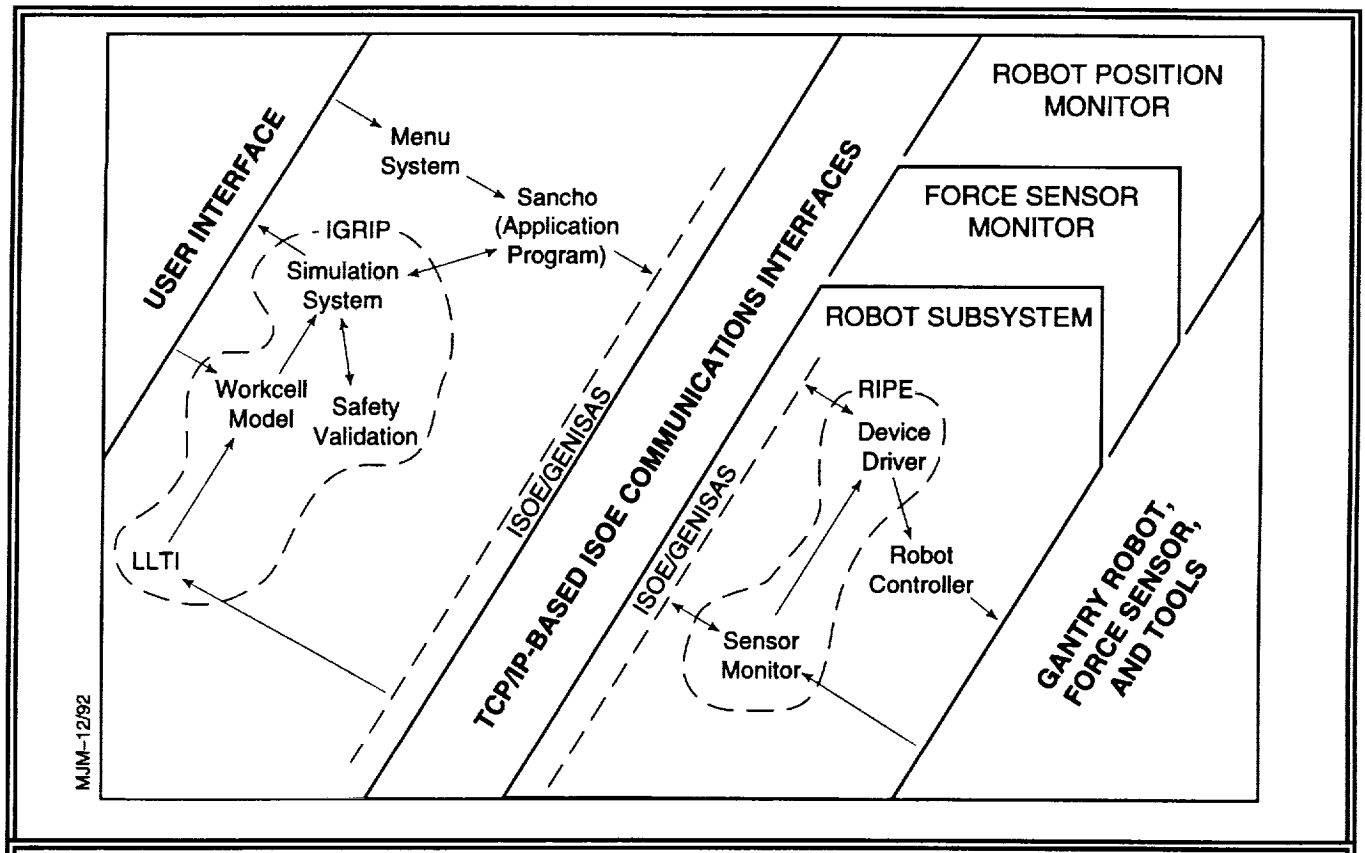

Figure 5: General Diagram of Graphical Programming System Using External Control Feature of IGRIP

approach also limited Sandia's ability to convert Supervisors to use other simulation systems.

Recently, we have leveraged features of IGRIP that allow us to write the application portion of our Graphical Programming Supervisory systems external to IGRIP while retaining IGRIP as the simulation system. The feature that allows this external control of IGRIP is called Socket-mode in IGRIP and uses a standard TCP/IP communications interface to form the link. We have surveyed several available robot simulation systems and have found that they also contain the ability to be controlled from external programs in similar ways.

We are currently writing a Supervisor Application Program called Sancho and an interface library which accesses IGRIP through socket mode. We are designing Sancho and the library to allow it to be rewritten to access other simulation systems. This will allow Sandia and other system developers to rapidly reuse supervisor software on multiple simulation packages and robotic systems.

Sancho uses Unix-based menu systems, operating system services, and communications systems and is directly compatible with ISOE and GENISAS. We are constructing our interface library between IGRIP and Sancho to allow us to use other simulation systems by changing the interface library. We are exploring methods to integrate other advanced technology including path planning and advanced sensor fusion by developing communications-based interfaces to the new subsystems.

Figure 4 shows a prototypical example of a Graphical Programming system that was written with Sancho. The robot in the figure is the gantry robot (see *Graphical Programming System Examples*) used for radiation surveys and the lines coming from the robot's tool show a path that the robot followed. The menus in this system use an X-windows menu system and Sancho is written in C. The menus allow the operator to command tasks that result in robot motion and also allow the operator to change viewing angles and other system parameters.

A generalized connection diagram of the new system, shown in Figure 5 and Figure 6, shows the detailed connection diagram for the first implementation of the system. Figure 5 shows how Sancho, the Application Program, is separate from the simulation system and Device Drivers and shows how

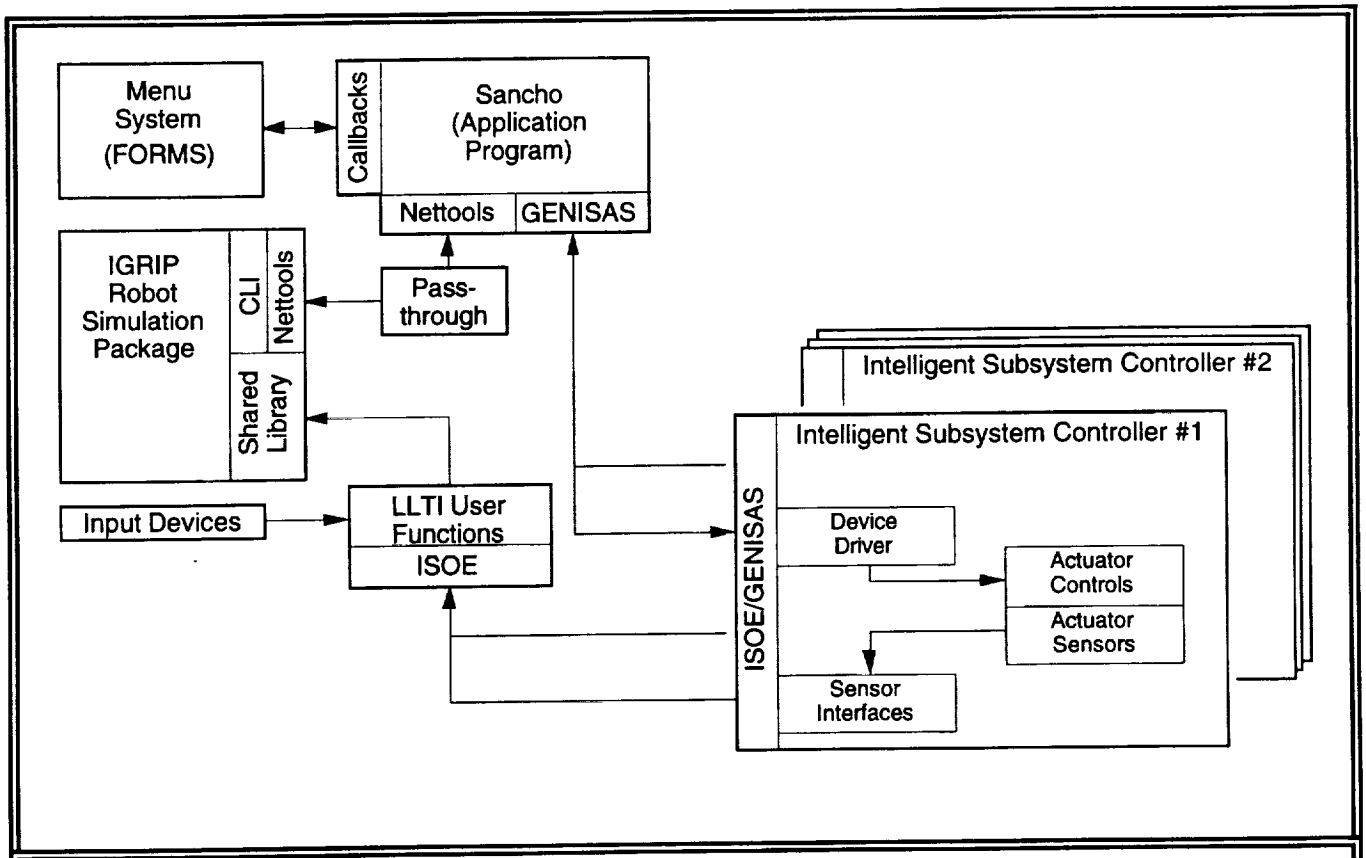

Figure 6: Prototype Graphical Programming System Using Sancho Application Program

the general structure of the new system corresponds to the GISC architecture.

Figure 6 shows how Sancho communicates Command Line Interface (CLI) commands to IGRIP through Nettools (a Deneb interface that uses TCP/IP) and device commands to the robots through GENISAS. The robot Device Drivers interpret commands from Sancho and command the robots to perform their motions. (For some robots, the Device Driver is an interface program that communicates with the robot's commercial controller, while in other robots, the Device Driver commands the robot servo systems directly.) The Device Drivers also monitor the robots' sensor values and communicate that data either back to Sancho through GENISAS or to IGRIP through the Low-Level Telerobotic Interface (LLTI) (a shared library interface that IGRIP provides for monitoring robots and input devices).

FUTURE WORK

Sandia develops prototype systems that are agile and flexible to meet pressing national needs. Sandia will use the experience derived in developing these prototype systems to help formulate specifications for systems that industry will produce. Sandia is particularly focused toward developing robot-control architectures that support the evolution to more autonomous systems in a way that makes advanced technology accessible to the end user.

Most of the robot systems that are proposed for hazardous operations will require multiple robots, controlled by multiple personnel, and sharing common workspaces. These robots will likely need to be commanded by different personnel to achieve the various goals inherent in complex systems. This model is extensible to space systems. For example, assembly currently done in space with a master/slave telerobot interacting with an astronaut could be done by a material handler robot and a second robot with a dexterous manipulator.

While the tasks are different, significant portions of the robot hardware will be shared between different personnel. For example, the various demonstration systems mentioned earlier showed that the same robots can be used for a wide variety of tasks. The demonstrations showed that material handling, material processing, and environment sensing operations that share manipulators can operate more

efficiently and effectively than is possible using unique manipulators for each task. Safely sharing hardware will require development of control architectures that can be safely accessed by many different supervisory programs while maintaining single point-of-control.

This working environment calls for telerobotic control architectures optimized for dynamically changing workspaces. Robots will need to check that other robots are not tasked to cross their paths before they can be commanded. Operators will need to request and gain control of robots, perform their tasks, and relinquish control. We plan to work toward developing systems that can safely operate in these environments and provide optimal use of the robot systems for the various tasks.

Currently, Sandia is working to extend graphical programming in several directions that will allow these complex systems to be effectively and safely used. As described earlier, we are refining our techniques of constructing Supervisors to generically access the simulation systems. We are also developing ways to integrate new and existing technologies by providing generic software interfaces to key technologies including sensor fusion and path and task planning. Finally, we are refining our software approach to multiple robot control and shared access control of robots. Our current work in these areas is described below.

Sandia is using communications-based approaches to separate the application program from the simulation systems in the Graphical Programming Supervisors. Decoupling the Supervisor application programs from the simulation systems will provide necessary experiences in understanding simulation system requirements independently of a particular vendor's current options. This experience will allow Sandia to help define achievable system specifications. It will also help Sandia to explore and suggest interface features for the simulation systems. Finally, it will help Sandia to pinpoint and implement features that will be required to safely bring advanced technologies to the users.

Sandia has recently begun projects to implement Graphical Programming with a second simulation system, and is supporting another RTDP member lab to implement Graphical Programming on a third simulation system. Lessons learned on the detailed implementation of these systems will support building a general interface specification to simulation systems. The result of this work will facilitate development of Supervisors that are independent of and portable between different simulation systems.

Sandia plans to integrate the results of its strong research program in path planning [15, 16, 17, 18, 19] into the GISC environment. Initially, path planners may be integrated through communications interfaces with the Supervisor's Application Program. Later, path planners may be integrated into commercial simulation systems. Our current research uses C-space models [20] for path planning because they provide computationally efficient workspace representations for planning collision-free motions. Path planners will need access to the dimensional database and will represent the dimensional information with unique internal representations that will be computed from the simulation system's geometric models. We are working to make that conversion process more practical.

Sandia also plans to integrate structured approaches to sensor fusion with Graphical Programming Supervisors. Sandia's MINILAB [21] system demonstrates that general architectures for sensor fusion are feasible, cost-effective solutions for integrating sensor information in the field. Recent technical advances in graphic display hardware, including texture mapping, make it possible to directly map and display sensor data on graphic surfaces in the simulation system. Simulation system software will soon be available to use these hardware capabilities.

New sensor fusion techniques will be extremely useful in robotic systems. For example, new hardware allows video and sensor-generated images to be mapped onto surfaces to let an operator accurately command a robot to reinspect areas identified in initial surveying operations. Volumetric-based data could be mapped onto surfaces to let the operator graphically locate *hot spots*, or be mapped onto critical parts of the robot to help the operator minimize dose counts to those parts. Surface-penetrating radar and other data could be mapped onto planes or *magic wands* that the operator would move through the model to better understand the environment. These sensor interfaces will improve efficiency by letting the operator directly command the robot to work on substances that would otherwise be invisible.

To better understand multiple-robot control, Sandia is developing several multiple-robot laboratories and the control software needed to control these robots concurrently. In one lab, Sandia has built a coarse/fine manipulator system from two separate robots to refine control techniques applicable to

robot systems like that used in the underground storage tank demonstration at Hanford. In another laboratory, Sandia is teaming a large pedestal robot (a Puma 762) with a gantry robot to explore telerobotic control strategies for robots that completely share their workspaces. A common interest in these systems will be in developing reusable supervisory software that can be applied across many applications. Sandia then plans to apply experiences gained in these two efforts to develop strategies for multi-operator control environments.

CONCLUSIONS

Sandia has developed and is refining Graphical Programming, an advanced robot control approach that uses visualization software to preview and monitor robot motions on a task-by-task basis. By using the Generic Intelligent System Controller approach and commercial visualization software, these systems are faster to implement and operate, safer to use, and cheaper overall than competing teleoperation or autonomous systems.

Recent systems development efforts have given Sandia a strong base of experience in extending graphical programming to a wide range of operations. Sandia is implementing the application programs for Graphical Programming Supervisors as separate programs that interact with the simulation software through a communications interface. This approach facilitates better software reuse and simulation system independence. New robot technologies (including advanced path planning and sensor fusion) are being integrated into Graphical Programming to further enhance operator efficiency without taking control away from the operator or adversely affecting operational safety.

Telerobotic servicing of space assets poses many challenges for robot control development. New Supervisory approaches are being developed to allow multiple robots to be easily controlled for cooperative tasks by a single operator. Other techniques are being developed to allow multiple operators to better share common resources. These control approaches will be needed to allow robots to safely, efficiently, and cost-effectively perform space servicing tasks.

ACKNOWLEDGMENTS

The work described in this paper would not be possible without the combined efforts of many people in the Sandia Intelligent Machine Systems Division. Specifically, we thank Lisa Desjarlais for IGRIP application programming assistance, Dave Miller and Charleene Lennox for building and supporting device drivers for the gantry and painting robot systems, Bill Davidson for his assistance in the communications interfaces, and Mike Griesmeyer for advice in generalizing the task control approach.

REFERENCES

1. Christensen, B. K., Drotning, W. D. and Thunborg, S., *Model Based, Sensor Directed Remediation of Underground Storage Tanks*, **ANS Annual Meeting, Remote Systems Technology**, Nashville, TN, June 10-14, 1990; *Robotics and Remote Systems*, M. Jamshidi, Ed., **Fourth ANS Topical Meeting on Robotics and Remote Systems**, Albuquerque, NM, February 1991.

2. Christensen, B. K. and Desjarlais, L. M. , *A Graphical Interface for Robotic Remediation of Underground Storage Tanks*, **First IEEE Conference on Visualization – Visualization '90**, San Francisco, CA, October 23-26, 1990.

3. Christensen, B. K., Drotning, W. D. and Thunborg, S., *Graphical Model Based Control of Intelligent Robot Systems*, **1991 IEEE International Conference on Systems, Man, and Cybernetics**, Sacramento, CA, May 1991.

4. Christensen, B. K., Griebenow, B. L. and Burks, B. L. , *Graphic Model Based Control of Robotic Systems for Waste Remediation*, **ANS Winter Conference**, San Francisco, CA, November 10-14, 1991.

5. Schenker, P. S., Kim, Won S. , Venema, S. C. , Bejczy, A. K. , *Fusing Human and Machine Skills for Remote Robotic Operations*, **Conference on Sensor Fusion III: 3-D Perception and Recognition**, Boston, MA, November 5-8, 1990. *Also* **Proceedings of SPIE – The International Society for Optical Engineering**, V 1383 1991, p 202-223.

6. Chiruvolu, Ravi, Hwang, V. S. , Sheridan, T. B. , *Virtual Display Aids for Teleoperation*, **Conference on Cooperative Intelligent Robotics in Space II**, Boston, MA, November 12-14, 1991. *Also* **Proceedings of SPIE – The International Society for Optical Engineering**, V 1612, 1992, p 299-310.

7. Quintero, Richard, ed., *DOE/NIST Workshop on Common Architectures for Robotic Systems*, **NIST Special Publication 784**, April 1990.

8. Quintero, Richard, ed., **Proceedings of the Second DOE/NIST Workshop on Common Architectures for Robotic Systems**, Seattle Washington, January 23-24, 1991. (Available from NIST, Robot Systems Division, Building 220, Rm B124, Gaithersburg, MD 20899).

9. Harrigan, R. W., Intelligent Automated Control of Robotic Systems for Environmental Restoration, **ANS Summer Meeting**, Boston, 1991.

10. Griesmeyer, J. M., McDonald, M. J. , Harrigan, R. W. , Butler, P. L. , and Rigdon, J. B. , **Generic Intelligent System Control (GISC),** Sandia Internal Report, SAND92-2159, October 1992.

11. Harrigan, R. W., *Automating the Operation of Robots in Hazardous Environments*, Planned for IROS 93 conference.

12. Miller, D. J., Lennox, R. C. , **An Object-Oriented Environment for Robot System Architectures**, *Control Systems,* IEEE, Vol. 11, No. 2, February 1991.

13. Mogal, Joshua S., *IGRIP – A Graphics Simulation Program For Workcell Layout And Off-Line Programming*, **Robots 10 – Conference Proceedings**, Chicago, IL, USA, 1986 April 20-24, Publ. by Robotics Int of SME, Dearborn, MI, USA p 10. 65-10. 77

14. **IGRIP User Manual**, Deneb Robotics Inc., 3285 Lapeer Rd. W. PO Box 214687, Auburn Hills, MI 48321-4687.

15. Chen, P. C. and Hwang, Y. K. , *Practical Path Planning Among Movable Obstacles*, 1991 **IEEE International Conference on Robotics and Automation**, Sacramento, CA, April 7-12, 1991.

16. Chen, P. C. and Hwang, Y. K. , *SANDROS: A Motion Planner with Performance Proportional to Task Difficulty*, **1992 IEEE International Conference on Robotics and Automation**, Nice, France, May 1992.

17. Chen, P. C., **Effective Path Planning Through Task Restriction, Sandia Report SAND91-1964**, Sandia National Laboratories, June 1992.

18. Chen, P. C., *Improving Path Planning with Learning*, **Ninth International Machine Learning Conference**, Aberdeen, Scotland, June 1992.

19. Hwang, Y. K. and Ahuja, N., *Gross Motion Planning — A Survey*, **ACM Computing Surveys**, vol. 24, no. 3, September 1992, pp. 219-292.

20. Donald, B. R. **Motion Planning with Six Degrees of Freedom**, Mass. Inst. Of Tech. Thesis, May 1984.

21. Feddema, J. T., A *Miniaturized Sensor System for In Situ Robotic Characterization of Hazardous Waste,* **Proceedings of Spectrum 92: Nuclear and Hazardous Waste Management,** pp 75-80, Boise, ID, August 23-27, 1992.