# Graphic-Simulator-Augmented Teleoperation System for Space Applications

Kazuo Machida,* Yoshitsugu Toda,* and Toshiaki Iwata†
*Electrotechnical Laboratory, Tsukuba, Japan*

**A teleoperation system augmented by a real-time graphic simulator has been proposed for efficient operation of a space telerobot and the laboratory model has been developed. In this system, an operator teaches task sequences to a slave arm by driving the three-dimensional graphic image of the arm on the simulator display, using a master arm. The sequences are stored and edited interactively, and the modified trajectories are transmitted to the slave arm. Flexible teleoperation is realized by forward/reverse reproduction of the path at various speeds, either on-line or off-line to the slave arm. In the simulator, arm motion and interaction with the work environment are computed in real time, and the animation of the task and the pseudoforces are fed back to the operator. This simulator is integrated with a slave-manipulator system designed for a vacuum environment and a master-manipulator system with a universal hand controller for space applications.**

## Introduction

A GRAPHIC simulator has the potential to play an important role in operating space telerobots for equipment assembly and fabrication of structures in orbit, since sufficient confirmation of task execution is difficult on the ground beforehand. The SIMFAC was developed to support the operation of the Shuttle remote manipulator system,[1] and the rate-controlled teleoperator task with time delay was examined,[2] using a real-time graphic simulator. This type of research is directed toward simulating manipulator motion on the display. Another approach studied is applying a graphic simulator to a teleoperation control loop directly, as a means of advancing operational capability. Our study belongs to this latter approach. Noyes and Sheridan[3] proposed a graphical predictor display to compensate the time delay for operating a manipulator by man-in-the-loop. Recently, Conway et al.[4] presented a new concept of forward simulation and predictor display augmented by "time and position clutches" to control the resulting manipulation paths and event transitions. At approximately the same time, we proposed the idea of task editing for efficient operation of a telerobot with a bilateral master-slave manipulator system using a real-time graphic simulator.[5] The graphic-simulator-augmented teleoperation system has been developed based on this concept and is presented in this paper. Although our approach is similar to Conway's concept, it is based on the analogy of editing with a video tape recorder or a word processor, in contrast to his analogy of a mobile control, such as a "time clutch." Moreover, a unilateral joystick is used for the input device in his study, whereas we deal with a bilateral master-slave manipulator system where the simulator requires the functions for a real-time interference check and force reflection.

## Concept of the Graphic-Simulator-Augmented Teleoperation System

The graphic-simulator-augmented teleoperation system, called "GSAT," is an advanced teleoperator with interactive task editing and forward/reverse reproducing functions using a real-time graphic simulator, enabling tasks to be performed safely and efficiently. This technology may be positioned at the midpoint of a line connecting a conventional teleoperator with an autonomous robot. The system is effective for telerobots that have both teleoperation and autonomous operation modes. Figure 1 shows the operational concept of the GSAT and Fig. 2 shows a block diagram of the developed system. An operator may use three different modes—teleoperation, simulator augmented, and autonomous— changing from one to the other according to the circumstances. Ordinarily, in unstructured environments, the operator controls the slave arm carefully using the master arm in the teleoperation mode. The simulator-agumented mode is used in the following situations: 1) pretrial of teleoperation; 2) man-intervened task planning for the fusion between teleoperation and autonomous operation; and 3) intervention to autonomous operation when a robot fails or is unable to manage a task. For the first situation, an operator can execute a task safely by considering and learning the procedures beforehand. For the second situation, an operator teaches, edits, and debugs the arm motion for the task using the simulator, and resulting command sequences are transmitted to the slave arm. This procedure can be applied to support the task-planning process of an autonomous robot with immature artificial intelligence. Therefore, the simulator-augmented mode can be used as a link between teleoperation and autonomous operation. For the third situation, control is passed to the operator from a robot when it is

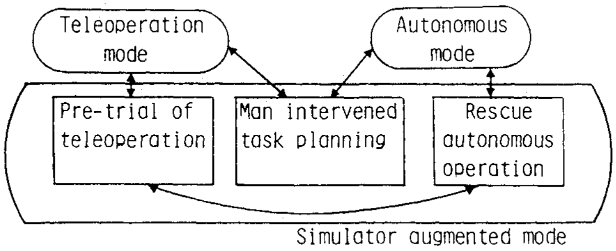

Teleoperation mode
Autonomous mode
Pre-trial of teleoperation
Man intervened task planning
Rescue autonomous operation
Simulator augmented mode

**Fig. 1 Conceptual relationship of the simulator-augmented mode with the teleoperation and autonomous modes.**

Presented as Paper 88-4095 at the AIAA Guidance, Navigation, and Control Conference, Minneapolis, MN, Aug. 15-17, 1988; received Oct. 29, 1988; revision received May 27, 1989. Copyright © 1988 American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Senior Research Staff, Frontier Technology Division, Space Technology Section.

†Research Staff, Frontier Technology Division, Space Technology Section.

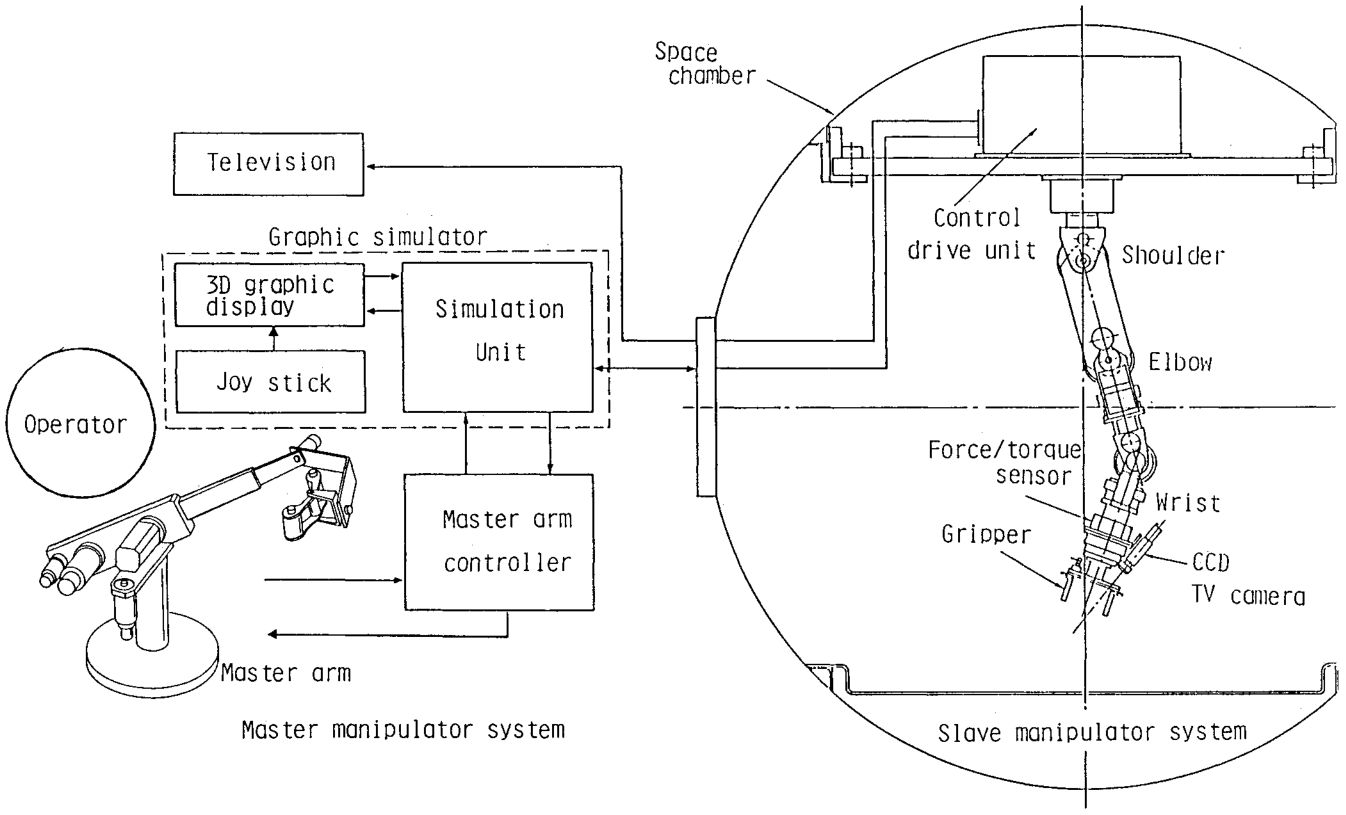


Fig. 2 Block diagram of the graphic-simulator-augmented teleoperation system.

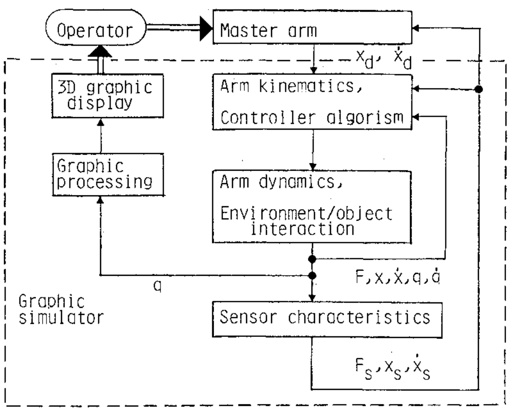


Fig. 3 Functional block diagram of the simulator.

unable to complete its task autonomously, via the simulator that monitors the operation. The operator rescues the robot by sophisticated remote control using the simulator, based on the monitoring and simulating information.

## System Description of GSAT

The developed graphic-simulator-augmented teleoperation system (GSAT) consists of a master-manipulator system, a slave-manipulator system, and a real-time graphic simulator, as shown in Fig. 2. The basic hardware-design philosophy is to provide the technology enabling adaptivity to space, which requires system durability in a vacuum, compactness, and extensibility. Universality is also emphasized in design, to be independent of the slave-arm configurations. A microprocessor-oriented architecture is employed for system extensibility. Each subsystem has one or more microprocessors and is loosely coupled by a standard RS-422A serial communication line, using an optical fiber cable. The slave-manipulator system was developed specifically to be used in a vacuum environment, and the master-manipulator system and the graphic simulator are designed to be used in a pressurized module of a space vehicle or space station, where unit compactness is required.

### Graphic Simulator

The basic functions required for a real-time simulator of a master-slave manipulator system are as follows: 1) slave-arm image display, i.e., image feedback; 2) force feedback; 3) arm-motion simulation; and 4) a man/simulator interface to provide flexibility in changing the models and parameters.

A functional block diagram of the simulator is shown in Fig. 3. An operator controls the slave-arm image, which is drawn on the three-dimensional graphic display, using the actural master arm. The main simulator design subjects are the hardware architecture and software algorithms for real-time simulation and the man/simulator interface to provide flexibility of operations. Concerning the first requirement, fast algorithms for dynamic analysis and interference checks must be introduced. The employment of a multimicroprocessor system shortens computation time by parallel processing and gives the system extensibility. Figure 4 shows the hardware block diagram of the simulator. An MC68000 microprocessor is employed as the kinematics central processing unit (CPU) that simulates the arm motion and executes interference check. Both an MC68020 and an MC68881 are employed for the graphics CPU, which generates graphics data for real-time simulation. The graphics CPU is also used as the host computer for computer aided design at the time of configuration modeling. The dynamics CPU, consisting of multimicroprocessors, is provided for simulating dynamic behavior of the teleoperator in a nongravitational field. The graphic data generated by the graphics CPU are transmitted to the three-dimensional graphics display through the general-purpose interface bus (GPIB), which has a transmission capability of 1 Mbyte/s. Although the three-dimensional graphic display employed here has $z$-buffers for hidden surface processing of the solid model, the animation cycle is insufficient for a solid-

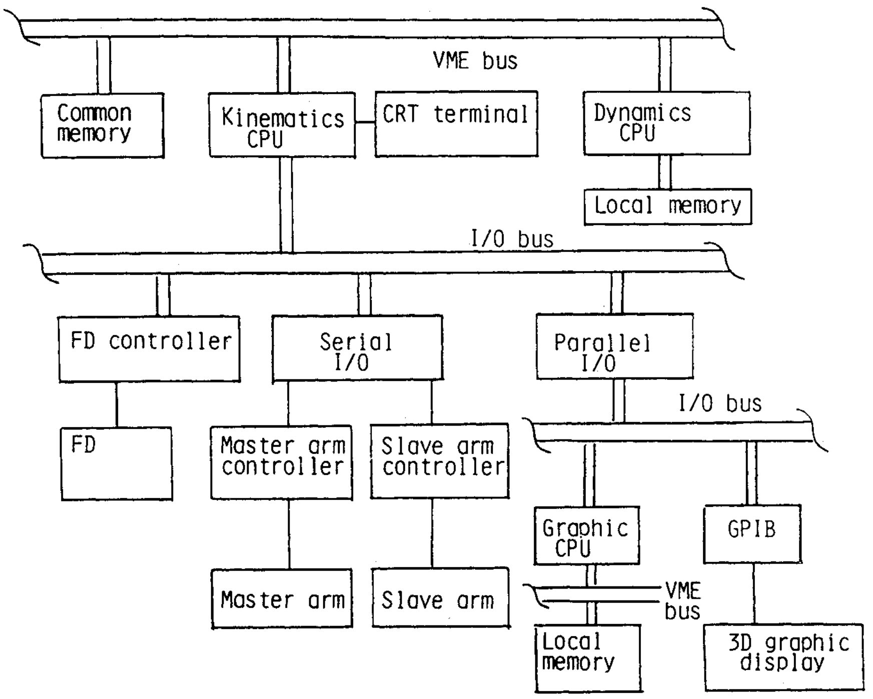


**Fig. 4 Hardware block diagram of the simulator.**

Table 1 Specifications of the slave-manipulator system

| Specifications | Magnitude |
|---|---|
| Bus voltage | 48 V |
| Input power | < 140 W |
| Arm degree of freedom | 6 |
| Arm length | 0.88 m |
| Tip force | 35 N |
| Grip force | 50 N |
| Control accuracy | |
| Position (PTP) | 0.5 mm |
| Rate: Translation | 5 mm/s |
| Rate: Rotation | 1 deg/s |
| Control mode | |
| Program: PTP | |
| Manual: Generalized bilateral | |
| Manual: Resolved motion rate | |
| Weight | |
| Arm | 12 kg |
| Control drive unit | 10 kg |
| Operation lifetime in vacuum | > 1000 h |

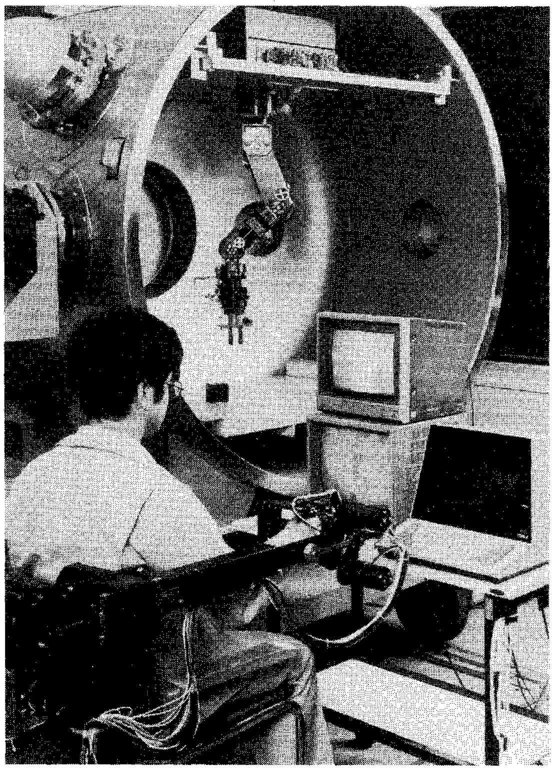


**Fig. 5 Slave-manipulator system installed in a vacuum chamber and the master-manipulator system.**

modeled manipulator. Therefore, we use a wire-frame model for the manipulator arm and use a solid model for the gripper and grapple rods where an operator most typically focuses his attention.

Concerning the man/simulator interface, CAD/CAE technology is applied to generate and change the configurations of the manipulator, the object, and the environment, interactively. The interference checking and the editing functions are described later.

### Slave-Manipulator System

The slave-manipulator system consists of a 1-m class articulated arm and a control drive unit (CDU). The system was designed to be adapted to a vacuum environment, and it has been operated in a vacuum chamber. The specifications are listed in Table 1. Figure 5 is a photograph of the slave arm installed in a vacuum chamber. Lightweight harmonic-drive actuators with a brushless servo motor, specifically developed for space use, are installed in the joints.[6] The manipulator, starting from the base, has a R-P-P-R-P-R joint configuration, where R and P denote rotation and pivot joints. A Sheinman-type six-dimensional force-torque sensor is integrated between the gripper and the last wrist roll joint to provide force and torque feedback to the operator. The grip force is detected by the load cells embedded in the finger, and it is reflected to the operator. A small charge-coupled device (CCD) TV camera is attached to the last joint of the wrist.

The CDU communicates with the master-manipulator system through a 300 kbit/s serial communication line. It controls the arm dynamics and supplies current to each actuator according to the commands. It also processes the data from the wrist sensor, grip sensor, and joint angle sensors, and transmits the data to the master-manipulator system. The software servo algorithm, which is based on the Paul inverse dynamic control method, is applied to control the wide range of payload mass under microgravity conditions. An MC68000 (12.5 MHz) microprocessor is employed for the CPU.

### Master-Manipulator System

The universal master-manipulator system that we developed is shown in Fig. 5. Universality is emphasized in designing the master-manipulator system enabling one master arm to operate various types of slave manipulators. This is desirable for space applications, because the pressurized room of a space vehicle is very valuable. This is also promising for the GSAT, since the simulator is essentially a general-purpose device. Universality is realized by transforming the kinematic relationship between a master and a slave arm.[7]

Configuration of the master arm is shown in Fig. 6. Position is established by three links near the base that form polar coordinates, and attitude is established by a three-axis gimbal mechanism, where the handle and the gripper are mounted. Force and torque are generated by a combinaton of brushless servo motors and pulleys. The neutral self-balance of the arm is realized by locating the actuators appropriately and by a counterbalancing mechanism installed in the prismatic joint no. 3. Microswitches are attached to the handle in order to change the control mode between either the bilateral position or a resolved motion rate control, and also to hold the current status of the slave manipulator for origin shift and work coordinate change, in one action. This idea of built-in switches is a great aid to flexible teleoperation of the slave arm. The coordinates and their scaling with respect to the work space are chosen arbitrarily by the operator. Figure 7 shows a control flow diagram of the master-slave system in bilateral-position control mode. The position and attitude of the hand-control point are computed from joint variables of the master arm, and are transmitted to the slave-manipulator system through a serial communication line, where attitude data are expressed by quaternions. In the slave-manipulator system, the parameters are transformed to the joint angles of the slave arm according to the inverse kinematic relation. Quaternions are more suitable for serial communication because transmission data are less than one half of direction cosine, and the amount of computation for transformations is also reduced by about 25%, thereby contributing to a faster control cycle. The strain data sensed by the force-torque sensor are processed and transmitted to the master-manipulator system, where the data are resolved into the master arm joints drive torques using the transpose matrix of the Jacobian. These transformations are executed by the MC68000 microprocessor boarded on each control system. The control-system program is described in C language and the numerical values are expressed in fixed-point format in order to shorten calculation time. A communication or control cycle of 40 Hz is obtained by these efforts for the master-slave manipulator system.

## Software Algorithm and Implementation

### Interference Check and Force Reflection

The most difficult problems for a real-time graphic simulator are realistic interference checking and force reflection in realtime. Although many algorithms have been proposed for

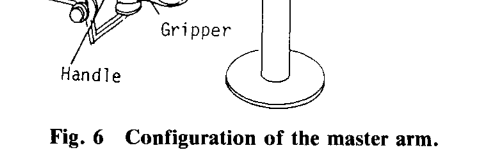


Fig. 6 Configuration of the master arm.

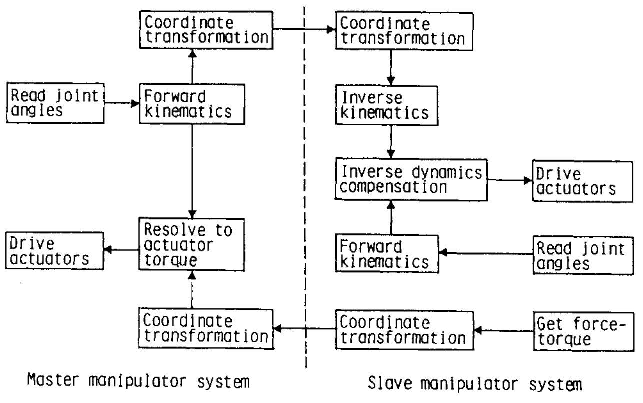

Coordinate transformation
Read joint angles
Forward kinematics
Drive actuators
Resolve to actuator torque
Coordinate transformation
Master manipulator system
Coordinate transformation
Inverse kinematics
Inverse dynamics compensation
Drive actuators
Forward kinematics
Read joint angles
Coordinate transformation
Get force-torque
Slave manipulator system

Fig. 7 Control flow diagram of the master-slave system.

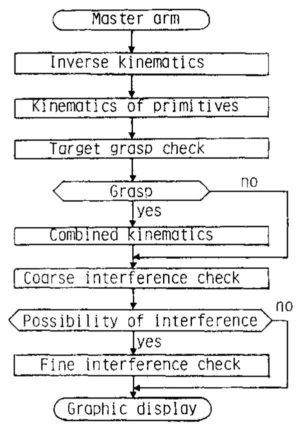


Fig. 8 Flowchart of the interference check.

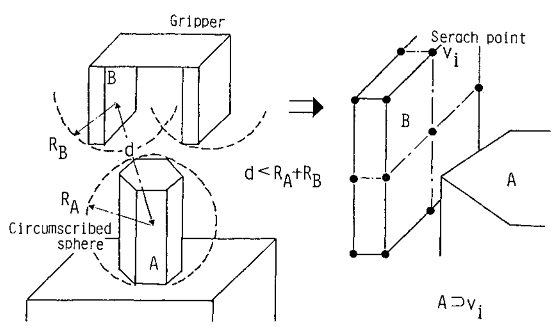
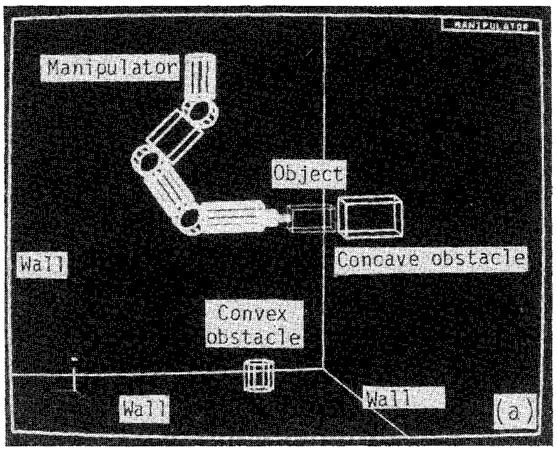


Fig. 9 Interference checking method for a gripper.

Table 2 Estimation of the computation numbers and times for the interference check

| Routine | Multiply | Add | Substitution | Conditional decision | Function call |
|---|---|---|---|---|---|
| Target grasp check | 450 | 800 | 430 | 80 | 99 |
| Combined kinematics | 100 | 80 | 80 | 10 | 4 |
| Coarse interference check | 130 | 180 | 160 | 30 | 16 |
| Fine interference check | 700 | 3200 | 920 | 220 | 220 |
| Total computation number | 1380 | 4260 | 1590 | 340 | 339 |
| Computation time, ms | 13.8 | 8.5 | 8.0 | 1.7 | 10.2 |
| Total computation time = 42.2 ms. | | | | | |



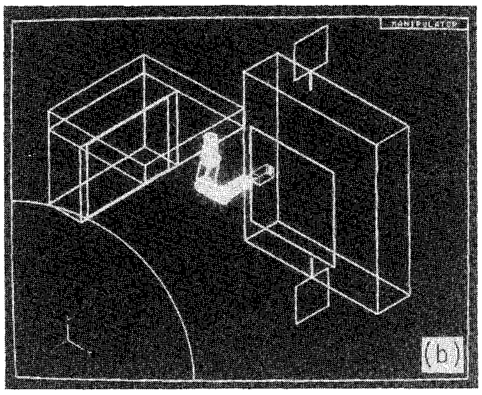


Fig. 10 Example of a graphic image.

Table 3 Measured computation time of simulator

| Routine | Time, ms |
|---|---|
| Kinematics CPU | **58** |
| Inverse kinematics | 10 |
| Kinematics of primitives | 6 |
| Target grasp check | 15 |
| Combined kinematics | 2 |
| Coarse interference check | 3 |
| Fine interference check | 15 |
| Communication | 7 |
| Graphic CPU | **88** |
| Communication | 3 |
| Graphic model | 85 |

interference checking in the field of CAD/CAM, they cannot be applied to the simulator, since they require too many computations to be simulated in realtime. Therefore, the configuration of the arm and the environment is expressed in a simplified form as long as it is suitable in practice, and a new approach with a two-stage strategy, a coarse and a fine interference check, is introduced to shorten the computation time. Figure 8 shows the algorithm concerned with the interference check. Position and attitude parameters transmitted from the master-manipulator system are transformed to the joint angles of the slave arm. The homogeneous transformations of the primatives expressing the manipulator are calculated. Figure 9 shows the method to check target grasping. The possibility of contact between objects and a gripper is examined coarsely by inclusion relations between both circumscribed spheres. If there is a possibility of contact, it is examined in detail to determine whether the points that are distributed on the edges of the gripper at the appropriate interval are inside the objects or not. The grasp condition is determined from the contact condition by referring to a table, where all possible cases of grasp are listed. If the grasp condition is satisfied, the kinematic expression of the manipulator combined with the object is generated. The grasp condition is checked for each cycle by comparing the current and former status. The possibility of interference between the manipulator and the obstacles is examined by checking the intersection between both circumscribed spheres, coarsely. If intersection occurs, a fine check that searches for containment of the points located on the edges of the polygon at the specified resolution is applied to the corresponding part. Although this method requires coordinate transformation of primitives of both the manipulator and the obstacle to each other, it is suitable for the simulator because the transformation matrix of the manipulator is computed each time to solve for arm motion.

The simulation cycle primarily depends on the fine interference check. A practical simulation cycle can be attained using a conventional microprocessor, if several limitations are imposed on the environment. We tried to simplify the models to obtain a computing time of less than 50 ms for interference checking, which might be required for real-time simulation. The computation numbers and processing times shown in Table 2 are estimated under the following conditions: 1) the configurations of objects and obstacles are restricted to rectangular or hexagonal pillars; 2) the resolution of the interference check is 1% of the arm length; and 3) the number of obstacles is limited to two. A computation time of less than 50 ms is estimated for the preceding conditions. The program is coded in C language and the fixed-point computation is also used to shorten the cycle time. The preceding simulation limitation is not critical because it covers the major parts of a teleoperator task, such as pick-and-place and insert-peg tasks. Figure 10a is an example of a display, which indicates that the manipulator uses a grapple rod to insert a rectangular parallelepiped object into the concave obstacle on the wall, where the contact point is displayed by blinking. Figure 10b is a display of a manipulator mounted on a space vehicle to service a satellite by replacing an orbit replacable unit (ORU). Table 3 shows the measured execution time to simulate each routine. The measured time for interference checking is almost the same as the estimated value. A smooth animation, refreshed every 88 ms, has been attained. Although the animation period is almost the upper limit for this system, which depends mainly on the capability of the graphic display, it is sufficient to supply real-time simulation for practical use.

The force is reflected to an operator by driving the master arm from information that is generated by the interference check. Although the direction and magnitude of the force is given by the simple model of the constant vector in the first version, it is very effective in giving the feeling of telepresence to an operator. Force reflection makes up for the lack of the reality of wire-frame images in teleoperation. In fact, it was demonstrated that the pseudoforce reflection effects considerably reduce the time for task teaching on the graphic simulator.

### Remote Task Teaching and Editing

Remote task teaching and task editing are key functions for augmenting the capability of the teleoperation system. The task editing function shown in Fig. 11 has been introduced. Command sequences and paths taught by an operator on the simulator graphic display are stored in buffer memories, edited interactively using the task editing module or task editor, and debugged. After desirable sequences or arm motions are obtained on the simulator, they are transmitted to the actual slave-manipulator system. The basic functions of the task editor are reverse/forward operations and overwrite/inserted operations. An operator can use the task editor interactively as with a video tape recorder or word processor. The path is reproduced forward or backward at any time rate, and an operator can interrupt in order to revise the path. The revision is executed by the master arm or program from the keyboard in the manner of overwriting or inserting. Ordinarily, off-line reverse/forward reproduction is used for editing the path, and on-line reproduction to the slave arm is used to execute the edited task. However, on-line reverse/forward reproduction can be used to execute sophisticated maneuvers by hybrid operation of direct remote control and edited-task execution. For example, the on-line reverse reproduction could be used to return a hung-up robot to the appropriate previous point for recovery, and the on-line slow forward reproduction could be used to execute a task for handling a large inertia payload slowly in space. As a matter of course, we can use it as the predictor display to compensate the communication time delay between the ground and a satellite, in the on-line real-time mode.

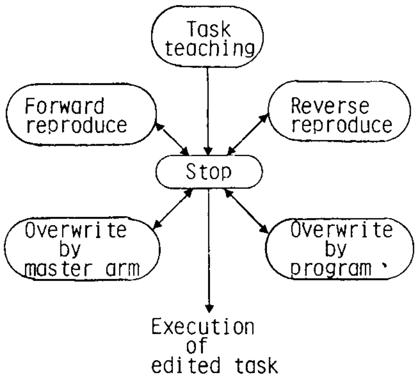


**Fig. 11 Function block diagram of a task-editing module.**

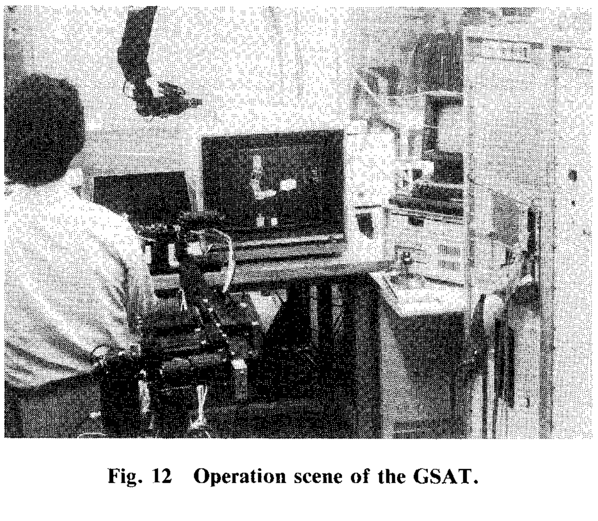


**Fig. 12 Operation scene of the GSAT.**

The system was integrated and these ideas were implemented. Figure 12 shows an operation scene of the GSAT. Many variations of the task were generated and executed efficiently, based on the reference task such as the ORU replacement, using the task editor. It was demonstrated that the GSAT has the potential for efficient and safe teleoperation in space. The present system cannot be applied to complex tasks that require more complex objects/environments and require stereoscopic images of solid models, because of insufficient processing speed. These problems should be settled in the near future with rapid progress in processor performance.

## Conclusions

A concept of graphic-simulator-augmented teleoperation has been proposed and the system developed, enabling tasks to be performed safely and efficiently in space. The real-time graphic simulator executes the interference checking algorithm with a two-stage strategy of coarse and fine search. A cycle time of about 50 ms was obtained with practical simplification of the models. An animation cycle of more than 10 Hz was attained. Pseudoforces are reflected to the operator by driving the master arm from the information that is generated by the interference check routine, and give a telepresence feeling to the operator. This simulator was integrated with a space-adaptive teleoperation system consisting of a slave-manipulator system designed for a vacuum environment and a master-manipulator system with a universal hand controller. The interactive task-editing module provides a powerful means for efficient and safe use of a teleoperated system. It has been demonstrated that the concept of GSAT is promising for future space telerobots, since it provides for the fusion technology between a teleoperator and an autonomous robot.

## Acknowledgment

The authors would like to acknowledge Mr. M. Kawachi, Mr. T. Nakamura, and Mrs. N. Arima of Mitsubishi Electric Corporation for their support.

## References

[1]McCulloguh, J. R., Sharp, A., and Doetsch, K. H., "The Role of the Real-Time Simulation Facility, SIMFAC, in the Design, Development, and Performance Verification of the Shuttle Remote Manipulator System (SRMS) with Man-in-the-Loop," NASA CP-2150, 1981, pp. 94-112.

[2]Pennington, J. E., "A Rate-Controlled Teleoperator Task with Simulated Transport Delay," NASA TM-85653, 1983.

[3]Noyes, M. and Sheridan, T. B., "A Novel Predictor for Telemanipulation through a Time Delay," *Proceedings of the Annual Conference on Manual Control,* NASA Ames Research Center, 1984.

[4]Conway, L., Volz, R., and Walker, M., "Tele-Autonomous System: Method and Architectures for Intermingling Autonomous and Telerobot Technology," *Proceedings of the IEEE International Robotic Conference,* Inst. of Electrical and Electronics Engineers, New York, 1987, pp. 1121-1130.

[5]Machida, K., Toda, Y., and Iwata, T., "A Real-Time Graphic Simulator for Manipulator and its Applications (in Japanese)," *Proceedings of the 3rd Space Station Conference,* Tokyo, 1987.

[6]Machida, K., Toda, Y., and Inoue, M., "Research and Development of a Small-Sized Space Manipulator," American Astronautical Society Paper 85-660, 1985, pp. 481-494.

[7]Bejczy, A. K. and Salisbury, J. K., Jr., "Kinesthetic Coupling Between Operator and Remote Manipulator," ASME International Computer Technology Conference, San Francisco, CA 1980.