ROBOSIM, A SIMULATOR FOR ROBOTIC SYSTEMS

Elaine M. Hinman and Ken Fernandez, Ph.D.
Marshall Space Flight Center

George E. Cook, Ph.D.
Vanderbilt University
Nashville, Tennessee

ROBOSIM, a simulator for robotic systems, was developed by NASA to aid in the rapid prototyping of automation. ROBOSIM has allowed the development of improved robotic systems concepts for both earth-based and proposed on-orbit applications while significantly reducing development costs. In a cooperative effort with an area university, ROBOSIM has been further developed for use in the classroom as a safe and cost-effective way of allowing students to study robotic systems. Students have used ROBOSIM to study existing robotic systems and systems which they have designed in the classroom.

Since an advanced simulator/trainer of this type is beneficial to not only NASA projects and programs but industry and academia as well, NASA in the process of developing this technology for wider public use. In this paper we give an update on the simulator's new application areas, the improvements made to the simulator's design and current efforts to ensure the timely transfer of this technology.

INTRODUCTION

ROBOSIM is a computer graphic robotic simulator developed by NASA-MSFC to aid in rapid prototyping of automation concepts. Cooperative development of a commercial and classroom version of ROBOSIM has been carried out by MSFC and Vanderbilt. In addition, ROBOSIM has been used in graduate courses at Vanderbilt as a trainer in robotic design.

Several applications of ROBOSIM will be described in this paper. While these are not the only current uses, they do show the usefulness of ROBOSIM in diverse fields -- from spaceflight to manufacturing to surgical aid development.

In the course of making ROBOSIM accessible to the larger robotics simulation and development community, several ports have been made to other workstations. A couple of these are described in more detail below.

DEVELOPMENT

ROBOSIM was originally developed in a VAX environment and provided graphical output to TEKTRONIX 4014 terminals and Evans & Sutherland graphics terminals [3]. Ports to the GTI Poly 2000, HP350SRX workstation, InterPro 360, and Silicon Graphics IRIS machine also exist [1, 2]. The TEKTRONIX terminal provides a low-cost and quick way to see the robot and workcell during the creation process, while the other systems allow real-time or near real-time viewing of robot motions.

To use the original ROBOSIM, the user created shapes and interconnecting joints using the ROBOSIM programming language. once a robot was modelled, motions could be generated either by user supplied routines or by using the default control algorithms [4]. The ROBOSIM program structure is shown in Figure 1.

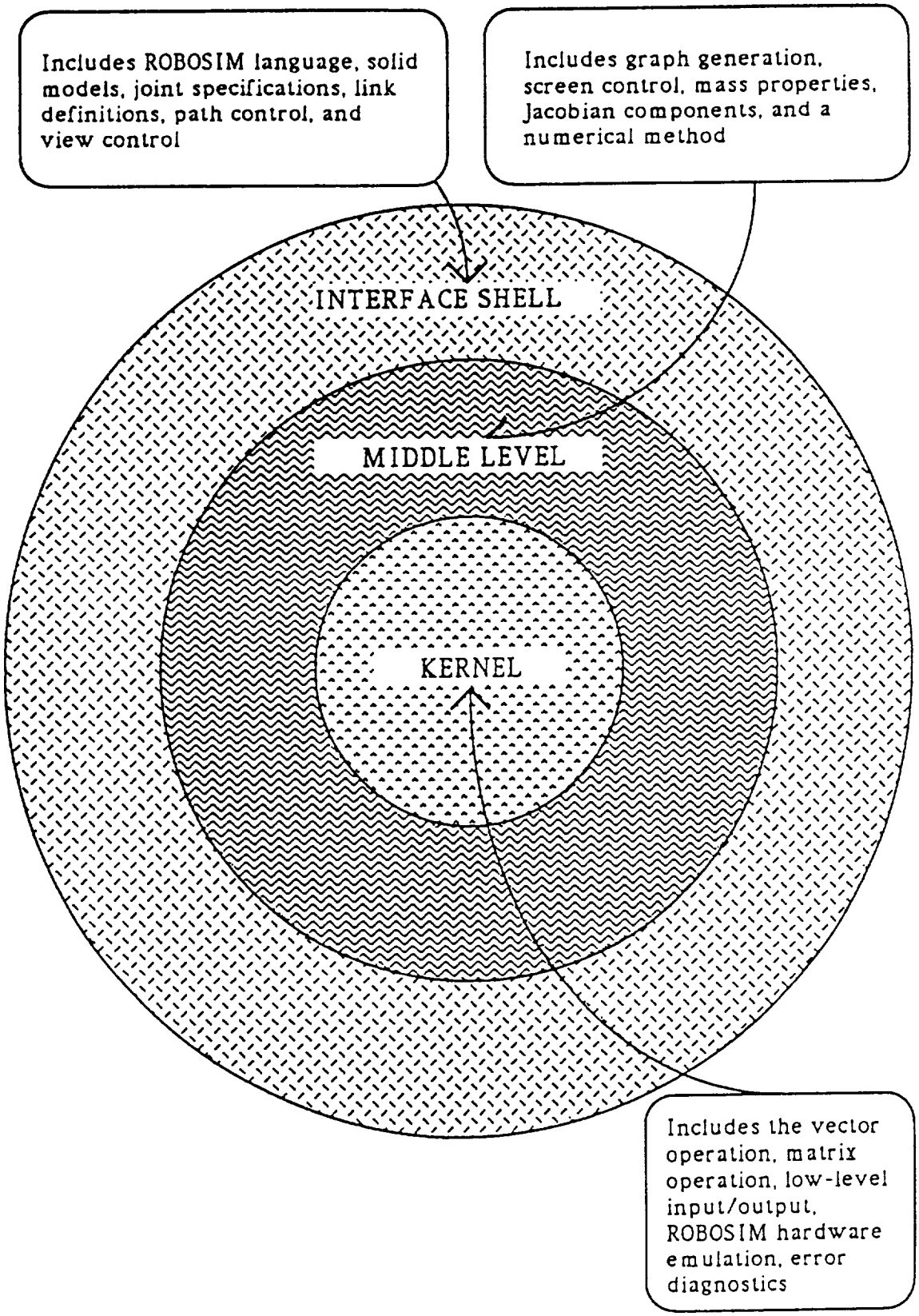

FIGURE 1. Tri-level structure of ROBOSIM

PORTS

The advent of high speed graphics workstations provided the impetus to port ROBOSIM to such a workstation [6]. However, to remain compatible with the VAX version of ROBOSIM, the VAX version has been kept as the kernel, with various features added to the front and back ends.

The TEKTRONIX 4014 terminal and Evans & Sutherland PS330 graphics system only allowed wireframe representations of the robot and workcell. The second implementation of ROBOSIM was on an HP350SRX graphics workstation. This allowed solid graphics, faster screen access, and facilitated the development of the R2 graphical interface discussed in the next section. The Starbase graphics library and X-Windows capabilities available on this workstation were used in the implementation of pull down menus and shaded graphics in the new ROBOSIM user interface.

A parallel development has been to integrate ROBOSIM onto the Intergraph InterPro 360 graphics workstation [2]. The Intergraph windowing environment, Environ V, was used in interfacing ROBOSIM with the InterPro workstation. Again, the foundation of ROBOSIM remained the same, however, the ROBOSIM-created graphical libraries were generated using Intergraph's graphic capabilities, and displayed in an Environ V window.

GRAPHICAL EDITOR

Once ROBOSIM was ported to the HP workstation, the 3D graphical editor, R2, was developed to provide an easier and more flexible interface to ROBOSIM [3]. The ROBOSIM kernel handles the rudimentary simulation tasks including the generation of primitive solids and rotation and translation operations performed on those objects. When modelling using the ROBOSIM programming language, the user was required to keep track of each object's dimensions, position and orientation with respect to other objects it may be connected to. In R2, menus and a mouse are used to generate the link primitives and to define the interconnecting joints. Since R2 is a graphical editor, the user may quickly and easily see and modify the robot workcell as it is being constructed. However, complete compatibility with ROBOSIM is maintained by having R2 generate ROBOSIM code. This way the capabilities of the original ROBOSIM are maintained while the enhancements of R2 are provided. Now, the user can design robots in the ROBOSIM code, by using R2, or through the use of custom programs.

SIMULATION LIBRARY

The following description of the ROBOSIM simulation library is from the paper by Springfield, Cook, Anderson, and Fernandez [3]. The library was first implemented on the HP-9000 workstation equipped with an SRX graphics accelerator.

The simulation library and environment provides methods to access the data structures created by ROBOSIM. The robots and other objects are specified and loaded into memory. These structures remain resident in memory while the simulation is running. The library provides an interface to these structures so that the user does not have to understand what is happening at that level. The library provides higher level facilities much like an actual robot programming language.

The simulation package allows one to use the robots that have been designed. The package consists of a library of C functions that operate on the files created by ROBOSIM. Although this package is far from complete, it allows simple simulations to be run. It also provides a framework in which to test the major components for the simulator: collision detection and dynamics. Having the simulator as a library of C routines allows more flexible methods for running simulations. Very specific and efficient simulations can be written in C which call the simulation functions directly. However, even at this level, much of the internal data structure is hidden from the user. This level of programming roughly corresponds to programming a robot in its programming language. For instance, one can tell a robot to move along a straight line or move a particular joint. Using these same routines, a very flexible, user-friendly interface can be built up, allowing an

interactive way to do simulations that are not too complicated, or that do not require great speed.

ROBOSIM provides most of the information required by the simulator through the files it creates. However, some information is not directly provided, but it can be determined from what is there. This involves the information required by the collision detection algorithm. ROBOSIM provides the Denavit-Hartenberg parameters, the A-matrix, the pseudo-inertia matrix, and a list of points which describe the physical structure of the robot. The internal data structure also includes areas that are not currently used, but will be at a later time. These include minimum and maximum joint angles, velocities, and accelerations. The structure also includes information related to the graphics display. The simulation package acts as intermediary between the user and the internal representation.

The simulation program that the user writes can turn on collision detection, request solutions to inverse kinematics problems, and display results graphically. The user can use the general numerical Jacobian method for inverse kinematics or provide an exact solution for the robot. The user simply passes the address of the function to the simulator, and the simulator will then use that function when solving inverse kinematics for that robot. A proposed extension to the simulator will allow the recognition of the possible robot configurations for which exact solutions exist. The exact solutions to these configurations would then be used instead of a numerical method, freeing the user from having to solve and code it himself. Several uses of the simulations system can be found in later sections.

Collision detection is very important in simulation of robots. one usually wants to know if the robot has collided with its environment or with itself. The collision detection algorithm implemented here is very similar to the POCODA (Polygon Collision Detection Algorithm) algorithm given in Scott E. Walter's dissertation from Cornell [7]. The collision detection algorithm has only two weak points. It does not handle concave polygons, and it will not signal a collision if one object is completely inside of another. The stipulation concerning concave polygons is not serious. ROBOSIM does not generate concave polygons unless they are the result of a custom object. Although R2 does not check for concave polygons, this feature could be implemented. In fact, algorithms exist to split concave polygons into convex polygons. Either of these features could be implemented fairly simply. The problem of not detecting a collision if one object is completely inside of another derives from the fact that the algorithm used is a polygonal collision detection algorithm and not a solid object one. However, assuming two objects start off outside of each other and movements are sufficiently small, then this should not prove to be a problem. This condition also prevents the ability of one object to pass through another (i.e. a movement is large enough that two objects do not overlap at any point). This algorithm does not detect collisions in the volume swept by an object moving between positions with another object, but rather only overlap of the objects at the starting and ending positions. But, if the distance between the positions is smaller than the smallest object, then there should be no problems.

APPLICATIONS

ROBOSIM has been used in several different applications. As a rapid prototyping robotic simulation system it is especially applicable when designing custom robotics for specific tasks. However, standard industry robots can also be modelled. Control algorithm development, positioning and task viewing studies, and workcell configuration are tasks for which ROBOSIM has been used.

CLASSROOM

ROBOSIM is used in an introductory graduate-level robotics class at Vanderbilt University. It is useful for aiding the students in visualizing transformations and perspective operations applied to items they have designed themselves. They can build objects and link them into kinematic chains. Once a robot is designed, the inverse kinematics can be solved and various control algorithms tested in graphics. Using graphic simulation, students have access to robotics development tools without the expense of providing robotics hardware for a classroom/laboratory environment.

SURGICAL POSITIONER DEVELOPMENT

ROBOSIM has been used by Dr. Bob Galloway and Allison Balogh of Vanderbilt to aid in development of a surgical positioner for use in brain surgery [3]. The positioner would be used to determine points of entry into the brain. Currently this is done by precomputing the location of the points with respect to an external reference. Greater flexibility would be provided by having a way to immediately see these positions. The current research is to see if a robot of sufficient accuracy can be built. ROBOSIM has been used to design the positioning arm and study its work envelope. The robot needs to be able to reach all positions on the head, while avoiding collisions with it. Currently, different positioner arm configurations are being studied.

WELDING

ROBOSIM has been used in a couple of welding simulation applications. one was in the development of a control algorithm for robotic welding of a Space Station airlock [2]. This application was developed on the InterPro workstation. A wireframe representation from this application is shown in Figure 2.

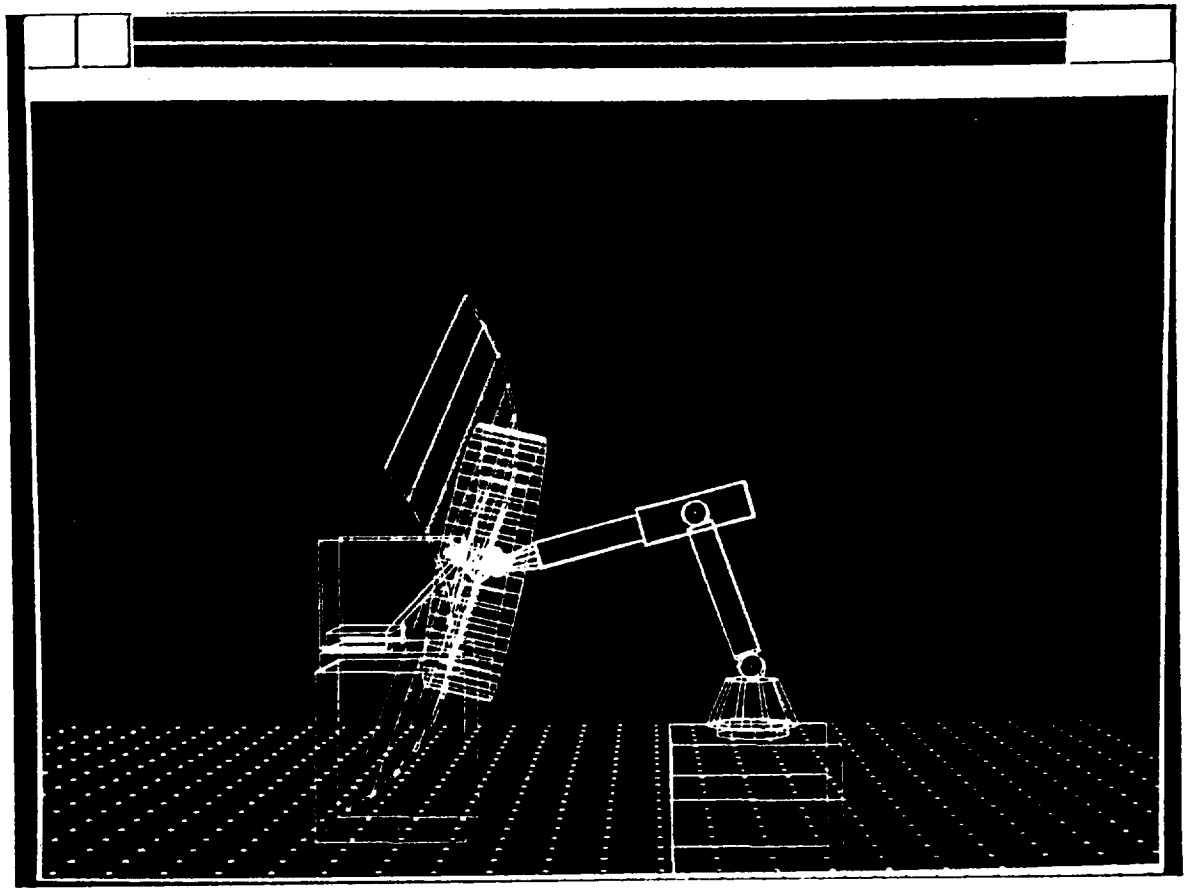

FIGURE 2. Airlock and Robot during weld.

Another welding application was in the testing and simulation of a downhand welding algorithm [5,8]. In this case a six degree-of-freedom robot performs the welding while a two degree-of-freedom positioner holds the workpiece. One implementation of this algorithm used a 2 degree-of-freedom table as the positioner, while another implementation used a PUMA 560 with 4 locked degrees of freedom as the positioner. In both cases, the object is to keep the weld feed in the downhand position to avoid pooling of the field. Based on the status of objects in the workcell, path information is generated and simulated by ROBOSIM in order to perform the weld. Joint constraints and collision detection are included in the path generation. Once a path is found, it may be sent to the actual robots and executed.

CONCLUSIONS

The previous examples point out a few of the areas in which the use of ROBOSIM is beneficial. Although ROBOSIM was originally developed at NASA for use in space robotics, it is useful to the larger robotics community. The commercial version currently under development will include the enhancements made in the second and later implementations of ROBOSIM. In this version, X-Windows may be used by multiple users to develop their applications. Although the processing speed of a workstation will be required for real-time and near-real-time simulation runs, workcell development can be carried out at a terminal through the common X-Windows interface.

In closing, ROBOSIM is a useful tool for robotics simulation in academia, government and industry.

ACKNOWLEDGEMENTS

The authors would like to express their appreciation to the Technology Utilization Office at MSFC, and in particular to Mr. Ismail Akbay for his support of this work.

REFERENCES

[1] Fernandez, K., R., and E. M. Hinman, "The Use of Computer Graphic Simulation in the Development of On-Orbit Tele-Robotic Systems," Proceedings of SPIE--The International Society for Optical Engineering, Advances in Intelligent Robotics Systems, 1986.

[2] Wilson, S., L. , Interfacing of a Robot Simulation Program with Graphic Utilities of an Intergraph Interpro 360 System, Master's Thesis, Vanderbilt University, 1990.

[3] Springfield, J., G. E. Cook, K. Andersen, and K. R. Fernandez, "ROBOSIM: A Simulation Package for Robots," University Programs in Computer-Aided Engineering, Design, and Manufacturing, ASCE Seventh Annual Conference, July 23-26, 1989.

[4] Fernandez, K., R., "ROBOSIM User's Guide," Internal Publication of the Electrical Engineering Department, Vanderbilt University, Nashville, TN, July 1986.

[5] Fernandez, K., R., Robotic Simulation and a Method for Jacobian Control of a Redundant Mechanism with Imbedded Constraints, Ph.D. Dissertation, Vanderbilt university, 1988.

[6] Springfield, J., ROBOSIM Workstation Extensions, Master's Thesis, Vanderbilt University, 1988.

[7] Walter, Scott Edward, Polygonal Collision Algorithm, Ph.D. Dissertation, Cornell University, 1985.

[8] Fernandez, K., R., and G. E. Cook, "Computer Graphic Simulation of an Algorithm for Controlling Downhand Position in Robotic Welding," SME paper MS86-209, Presented at the SME Conference on Robotic Solutions in Aerospace Manufacturing, March 3-5, 1986.