Int. J. Engng Ed. Vol. 12, No. 3, pp. 172–181, 1996
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1996 TEMPUS Publications.

ROBOTSIM: A CAD Package for Design and Analysis of Robots*

ALI M. EYDGAHI
Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

ROBOTSIM is a robotics simulation software package primarily for educational applications to demonstrate the design and analysis of robots and workcell environments. It assists students in understanding and visualizing many fundamental concepts of robot kinematics, robot programming, robot modeling using simulation software, and robotic workcell design.

INTRODUCTION

IN THE past two decades, educational institutions including universities, colleges, vocational schools and high schools, have introduced courses in robotics, automation, computer-aided design (CAD) and computer-aided manufacturing (CAM). These courses have become increasingly popular in the curriculum of electrical, mechanical, and industrial engineering as well as other technology departments such as computer science.

To meet the demand of these courses, many textbooks have recently been written emphasizing many different aspects of robotics [1–8]. Most of the introductory robotics courses emphasize kinematics, dynamics and robot programming [1–4] while more advanced courses consider subjects such as sensor integration, vision and artificial intelligence as applied to robotics [5–8].

In order to enhance the understanding of these subjects, many schools offer a robotics laboratory as a part of their courses to provide hands-on experience programming actual robots and setting up robotic manufacturing workcell layouts. However, facilities in many of these robotics labs are limited to a small number and variety of robots due to the high price of obtaining and maintaining robot equipment. This often forces strict limits on the number of students who can participate in lab exercises and fails to give a true understanding of the wide variety of different robot designs and layouts being used in industry today.

Graphical simulation software for the analysis and design of robots and workcell environments offers a cost-effective alternative for schools with limited laboratory budgets [9, 10]. It also offers new educational benefits over the use of actual robotics equipment by serving as a teaching aid for many fundamental robotics concepts. Computer simulation is particularly useful for demonstrating three-dimensional link axes and the three dimensional movements of link coordinate frames which are difficult to visualize when presented using only a textbook and a blackboard. It also offers students a more realistic and stimulating media for learning about robotics.

The ROBOTSIM software package offers realistic graphics, animation and an interactive mouse-driven environment. It allows the user to rotate and zoom the workcell in real time so that it may be viewed from any perspective. Features which are not found on other packages include commands for the automatic calculation and display of a D-H table of kinematic parameters and a set of transformation matrices relating the base, the end effector and each of the robot link frames.

This paper describes the ROBOTSIM software, its features and how an actual robot, the Rhino SCARA, can be modeled with ROBOTSIM.

BACKGROUND

A robot arm is modeled as a chain of rigid bodies called links. Each link has a three-dimensional local coordinate frame associated with it. These coordinate frames are assigned according to the Denavit–Hartenburg (D–H) convention. The links are connected by flexible joints. Joints may be classified as either revolute or prismatic. Using the link coordinate frames, a table of kinematic parameters may be defined for each link above the base using the D–H convention [1–5]. These kinematic parameters are defined as the joint angle θ, the joint distance d, the link twist angle α, and the link length a. These kinematic parameters may then be used to derive a four-dimensional homogeneous transformation matrix for any pair of adjoining links. The transformation matrix represents the change in the three-dimensional coordinate frame of one link with respect to the coordinate frame of the next link and is a function of both the kinematic parameters and the joint positions of all the links and joints between them. The following transformation matrix transforms the coordinate frame of link i to the coordinate frame of link $(i-1)$:

*Accepted 1 November 1995.

$$ {}^{i}A_{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

ROBOTSIM OVERVIEW

ROBOTSIM is written for IBM PCs and compatibles running under DOS with EGA or VGA graphics and a mouse. ROBOTSIM includes a library containing many commonly used industrial and educational robots. The library also includes objects which can be placed in the workcell and manipulated by the robots. ROBOTSIM has a built-in CAD system for designing new robots and objects for the library. Actual robots may be modeled using technical specifications. New robot designs may also be created and tested. Robot programs are generated with a built-in text editor and the ROBOTSIM instruction set. Robot programs can be executed both on the simulated graphics image using real-time animation or on an actual robot using a Rhino Mark III type controller connected to the computer using an RS-232 serial communication interface. The majority of ROBOTSIM's commands may be accessed using a mouse. Figure 1 shows the flowchart of ROBOTSIM's menu structure.

ROBOTSIM displays its main menu at the top of the screen. When a choice is made from the main menu, a window containing all sub-menu commands appears at the right hand of the screen. The remaining area of the screen is the workspace. It is where all graphics images and robot program text appears. Menu selections can be made easily with the mouse or with the keyboard. The main menu consists of five choices: WORKCELL, PROGRAM, DESIGN, OPTIONS and HELP.

WORKCELL module

The WORKCELL main menu choice displays a wireframe image of a robot workcell, executes robot programs using real-time animation and is used to teach points for new robot programs. The three-dimensional color graphics image of the workcell may be rotated and zoomed in or out to be viewed from any angle. The user can choose any joint and see the movement of that joint on the simulated robot. If an actual robot using a Rhino Mark III type controller is connected to the computer using the serial communication interface, it will move in synchronization with its simulated image. As the robot moves, its position in motor encoder increments is updated and displayed. WORKCELL has eight external inputs and outputs which can be used to allow ROBOTSIM to input data about the workcell's conditions and to control external devices. Input and output conditions are updated and displayed in real time. The TEACH command in the WORKCELL submenu allows a point label to be assigned to a particular set of robot joint motor encoder values. These taught points may be used later to create a robot program using the PROGRAM main menu choice.

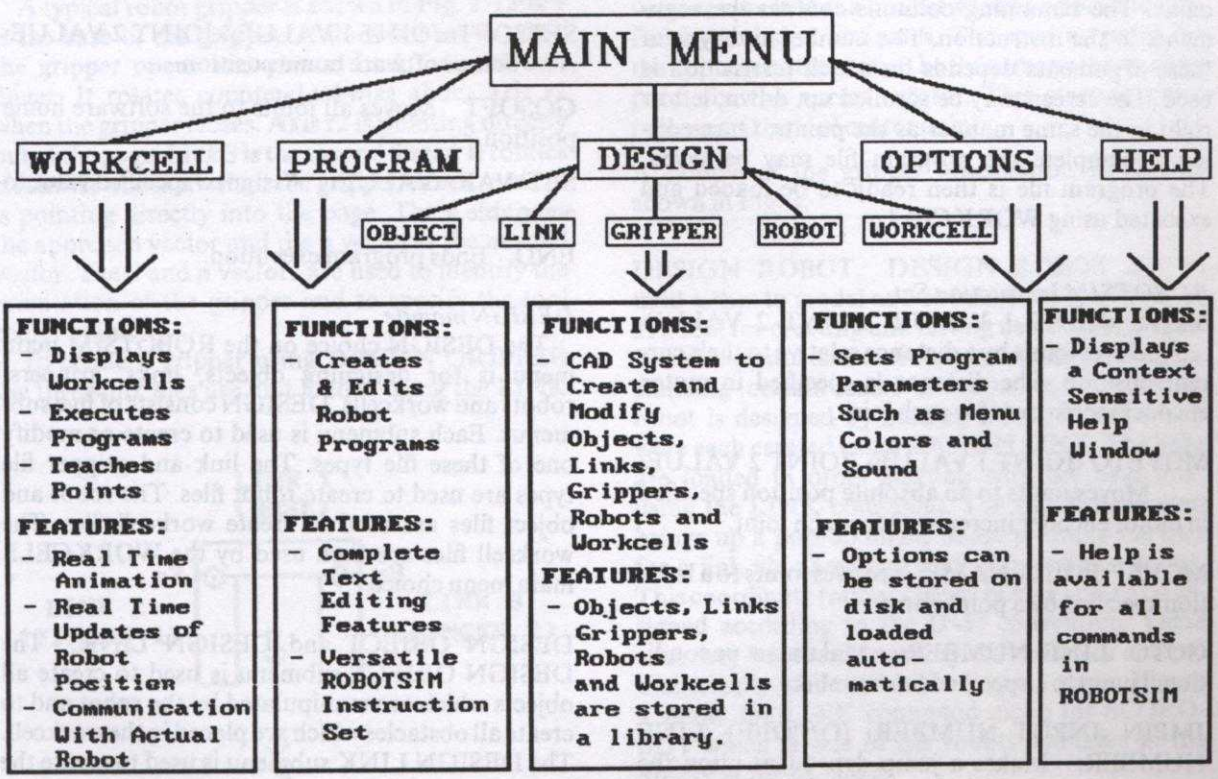

Fig. 1. ROBOTSIM menu structure.

PROGRAM module

The PROGRAM command on the ROBOTSIM main menu is for creating and editing robot programs. ROBOTSIM includes a complete instruction set of robot programming commands based on commonly used industrial robot programming languages. The ROBOTSIM instruction set includes commands to move robot joints, commands to control program execution, commands to adjust the graphics image during program executing and commands to communicate with the external inputs and outputs. A robot program may also include a set of points taught using the TEACH command in WORKCELL.

A ROBOTSIM program file consists of two parts. The first part contains the instructions for the robot program, and the second part contains an optional set of taught points. The PROGRAM line editor allows both the program and the taught points to be modified at any time. Instructions and points may be copied, inserted, deleted, moved and modified. The editor may be toggled to display either the taught points or the program commands. When the taught points are displayed the left-most column contains the point name. The following columns initially numbered 1–8 correspond to the position of the first eight joints for that point. The screen may be scrolled to the right to see the positions of additional joints. If >20 points exist, the screen may be scrolled up or down to see these additional points. Scrolling is performed using Page Up, Page Down, Page Left and Page Right keys or by using the arrow keys. When the instructions are displayed, the left-most column contains the ROBOTSIM instruction name. The remaining columns contain the arguments to the instruction. The number and type of these arguments depends on which instruction is used. The screen may be scrolled up, down, left or right in the same manner as the points. Once editing is complete the program file may be saved. The program file is then ready to be loaded and executed using WORKCELL.

ROBOTSIM Instruction Set

MOVE ‹JOINT 1 VALUE› ‹JOINT 2 VALUE› ... Moves joints by a distance relative to their current position. The distance is specified in motor encoder increments for each joint.

MOVETO ‹JOINT 1 VALUE› ‹JOINT 2 VALUE› ... Moves joints to an absolute position specified in motor encoder increments for each joint.

MOVEP ‹POINT NAME› Moves joints to a location specified by a point name.

GOTO ‹LINE NUMBER› Makes an unconditional jump to a specified line number.

JMPIN ‹INPUT NUMBER› (ON¦OFF) ‹LINE NUMBER› Makes a jump dependent upon the value of a specified variable.

OUTSIG ‹OUTPUT NUMBER› (ON¦OFF) Sets a specified output condition.

JMPVAR ‹VAR NAME› (›¦‹¦=) ‹VALUE› ‹LINE NUMBER› Makes a jump dependent upon the value of a specified variable.

GOSUB ‹LINE NUMBER› Calls a subroutine.

RETURN Returns from a subroutine.

WAITFOR ‹INPUT NUMBER› (ON¦OFF) Pauses program execution until a specified input condition is satisfied.ADD:

ADD ‹VAR NAME› ‹VALUE› Adds a specified value to a variable. Subtraction may be performed by using a negative value.

RESET Resets the workcell to its original position. It is the equivalent of the RESET command on the WORKCELL menu.

WINDOW ‹X VALUE› ‹Y VALUE› ‹Z VALUE› Changes the viewing window on the workcell. It is the equivalent of the WINDOW command on the WORKCELL menu. The workcell is shifted in the x, y and z directions as specified by the arguments.

VIEW ‹X VALUE› ‹Y VALUE› ‹Z VALUE› Changes the view position on the workcell. It is the equivalent of the VIEWPOINT command on the WORKCELL menu. The image is rotated in the x, y and z directions as specified by the arguments.

DELAY ‹TIME IN SECONDS› Causes a delay in program execution.

HRDHOME Returns all joints to the zero position.

SETSOFT ‹JOINT 1 VALUE› ‹JOINT 2 VALUE› ... Sets a software home position.

GOSOFT Moves all joints to the software home position.

SET ‹VAR› ‹VALUE› Assigns a specified value to a variable.

END Ends program execution.

DESIGN module

The DESIGN choice on the ROBOTSIM main menu is for designing objects, links, grippers, robots and workcells. DESIGN consists of five submenus. Each submenu is used to create or modify one of these file types. The link and gripper file types are used to create robot files. The robot and object files are used to create workcell files. The workcell files are then used by the WORKCELL main menu choice.

DESIGN OBJECT and DESIGN LINK. The DESIGN OBJECT submenu is used to create all objects which are manipulated by the robot and to create all obstacles which are placed in the workcell. The DESIGN LINK submenu is used to create the individual rigid links of a robot. The commands

for DESIGN OBJECT and DESIGN LINK are exactly the same, but the way in which the files are used after they are created differs. An object file is created or modified using DESIGN OBJECT and a link file is created using DESIGN LINK. Objects and links are simply a collection of lines which comprise a wireframe image. To design an object or link the user must specify the two endpoints between each line in global x, y and z coordinates. Each line may be drawn in any one of 16 colors. There are two methods of drawing lines. The first method uses the mouse to specify the endpoints of a line. The second method enters the endpoints numerically using the EDIT and BOX commands. The EDIT command brings up a window allowing the x, y and z coordinates of the endpoints of any line to be changed numerically. It also allows the color of the line to be modified. The BOX command offers a shortcut to creating cubes and other rectangular solids. The BOX command allows cubes and other rectangular solids to be created simply by entering a single number for the left, right, top, bottom, front and back edges of the box. The cube or rectangular solid may then be reoriented using the ROTATE command.

DESIGN GRIPPER. The DESIGN GRIPPER submenu command is used for designing grippers. A gripper is created by loading three link files which were each created using DESIGN LINK. The first of these three links represents the base of the gripper and the last two links represent the two fingers of the gripper. Specific information must then be input to indicate how the fingers of the gripper move when the gripper opens and closes.

A typical robot gripper is shown in Fig. 2. Link 1 is the base of the gripper. It does not move when the gripper opens and closes. Link 2 is the first finger. It rotates counterclockwise about axis r2 when the gripper closes. Axis r2 is pointing directly out of the page. Link 3 is the second finger. It rotates clockwise about r3 when the gripper closes. Axis r3 is pointing directly into the page. The **a** vector is the approach vector and the **n** vector is the normal vector. The **a** and **n** vectors are used to identify the orientation of the gripper and to specify the tool center point.

To create a gripper using DESIGN GRIPPER the three link files are first loaded using the LINK command. Certain information is then needed to complete the gripper design. The INFO command uses a pop-up menu to input the gripper joint's joint letter, resolution, minimum and maximum range, and its r2, r3, **a** and **n** vectors. These values may be described as follows:

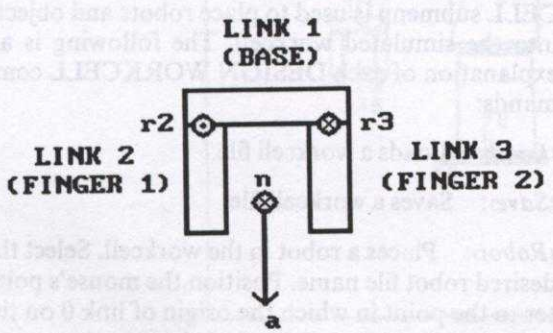

Fig. 2. A typical robot gripper.

‹*Joint Letter*›: Any letter (A–Z) which represents the gripper joint.

‹*Precision*›: The angle of rotation of the gripper fingers expressed in degrees for one motor encoder unit.

‹*Min Encoder Range*›: The minimum range of the gripper joint expressed in motor encoder units. This value represents the gripper open position and is typically zero.

‹*Max Encoder Range*›: The maximum range of the gripper joint expressed in motor encoder units. This value represents the gripper closed position and is typically the amount of rotation required to move the two fingers to the point at which they are about to touch one another.

‹*Rotational Axis Link 2*›: The location of rotational axis r2 expressed as a line between two points identified by global coordinates.

‹*Rotational Axis Link 3*›: The location of rotational axis r3 expressed as a line between two points identified by global coordinates.

‹*Normal Vector*›: The location of the normal vector expressed as a line between two points identified by global coordinates.

‹*Approach Vector*›: The location of the approach vector expressed as a line between two points identified by global coordinates. The starting point for the approach vector must be the same as the starting point for the normal vector. This starting point represents the tool center point.

A sample of the gripper information window is shown in Fig. 3.

DESIGN ROBOT. DESIGN ROBOT can be used either to model and simulate actual robots or as a tool for creating and testing new robot designs. It is also useful as a tool for visualizing and understanding certain basic concepts in robotics. A robot is designed by loading a set of links which were each created using DESIGN LINK. The links are loaded in order starting with the base link, using the LINK command. The LINK command brings up a pop-up menu which prompts the user for a link file name and a local coordinate frame. This coordinate frame consists of an x and z axis assigned according to the D–H convention. These axes may be either drawn by the mouse or entered numerically while inside the pop-up menu. Figure 4 shows the LINK command menu for SCOROBOT which is one of the robots available in the library of the package.

Once two or more links have been loaded and

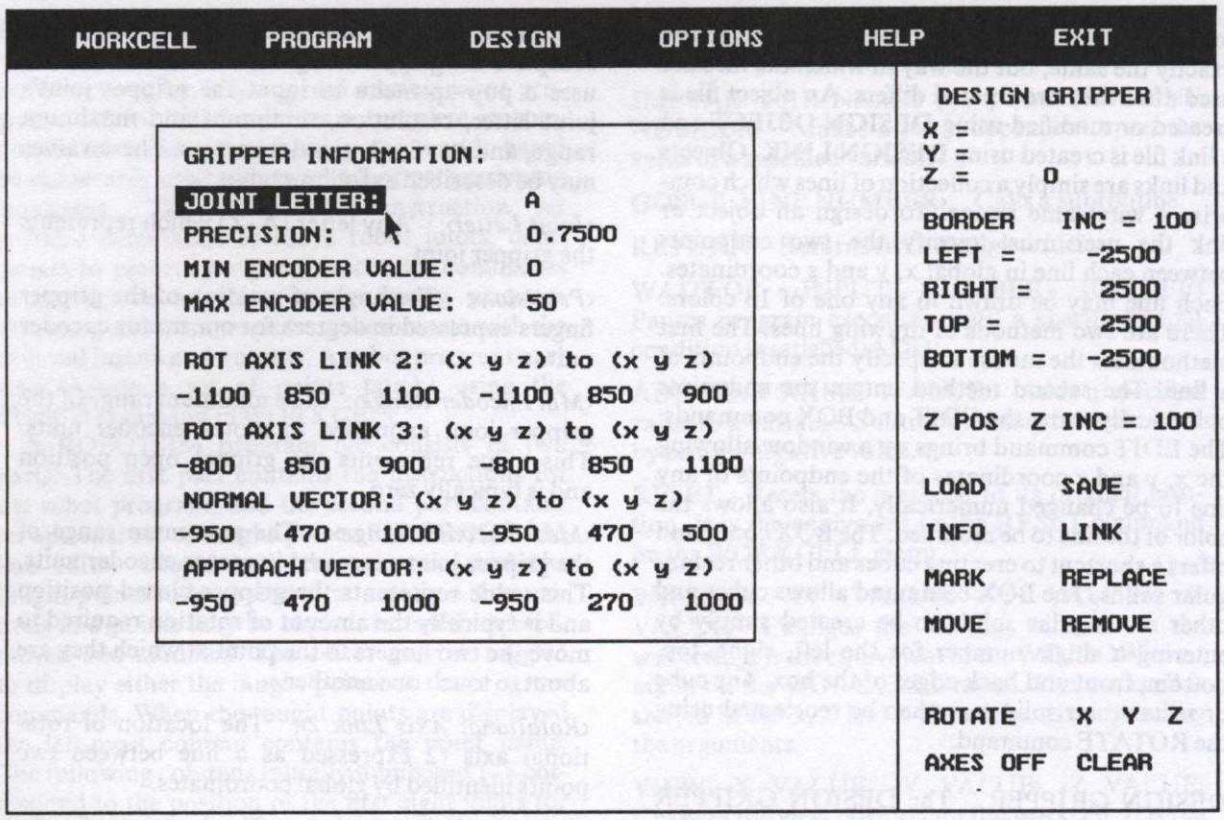

Fig. 3. A sample gripper information window.

assigned coordinate frames, the JOINT command may be used to specify the joint information. Each new joint which is created with the JOINT command is numbered starting from 1. Joint n affects link n and above for all n. The JOINT command brings up a pop-up menu which allows specific information about the joint to be entered. A sample of JOINT command window for SCOROBOT is shown in Fig. 5. The information required is as follows:

‹*Joint Type*›: Choose between revolute and prismatic.

‹*Joint Letter*›: Selects any letter (A–Z) which will identify the joint.

‹*Precision*›: The amount of rotation expressed in degrees for a revolute joint or the amount of translation expressed in global coordinates of one encoder unit for a prismatic joint.

‹*Min Encoder Range*›: The minimum joint range as expressed in motor encoder units.

‹*Max Encoder Range*›: The maximum joint range as expressed in motor encoder units.

‹*Coupling*›: Joint coupling occurs when one joint moves to counteract the motion of another. ROBOTSIM supports next joint coupling. This means that joint $n + 1$ will move to counteract the rotation or translation of joint n. Coupling may be set to ON or OFF.

Once each robot link and joint have been created, a robot gripper may be added using the GRIPPER command. The **n** and **a** vectors of the gripper become the x and z D–H coordinate axes for the end effector. The robot design is then complete and may be stored as a robot file to be placed in a workcell along with other robots and objects using DESIGN WORKCELL.

Once the robot has been designed, a table of kinematic parameters is automatically created. This table may be viewed with the TABLE command. The MATRIX command will display the transformation matrix for any two specified link numbers. Samples of TABLE and MATRIX commands for SCOROBOT are shown in Figs 6 and 7, respectively.

DESIGN WORKCELL. The DESIGN WORKCELL submenu is used to place robots and objects into the simulated workcell. The following is an explanation of each DESIGN WORKCELL commands:

‹*Load*›: Loads a workcell file.

‹*Save*›: Saves a workcell file.

‹*Robot*›: Places a robot in the workcell. Select the desired robot file name. Position the mouse's pointer to the point in which the origin of link 0 on the robot is to be placed. Press the left mouse button.

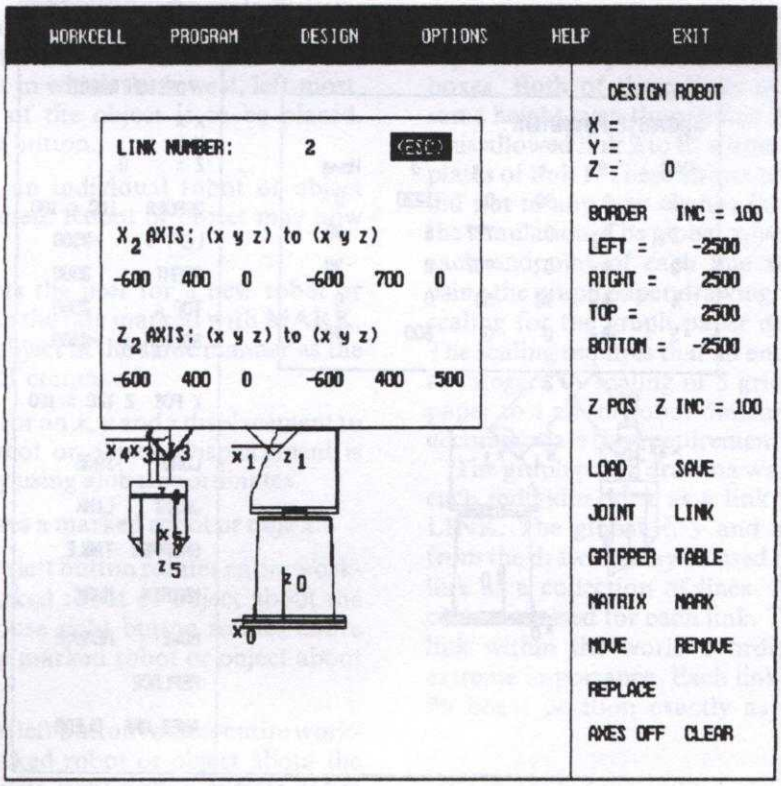

Fig. 4. A sample LINK command menu.

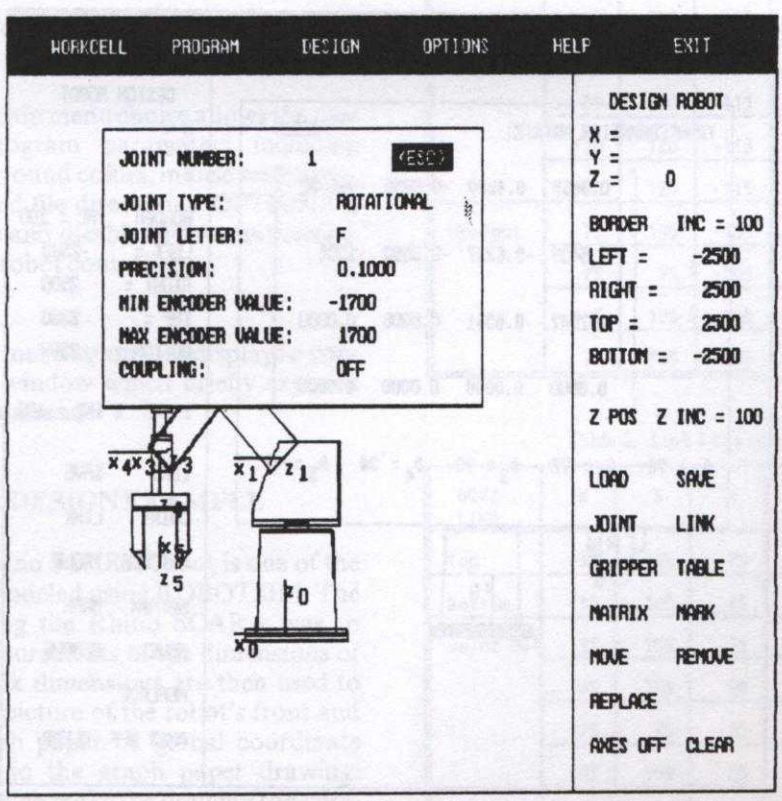

Fig. 5. A sample JOINT command window.

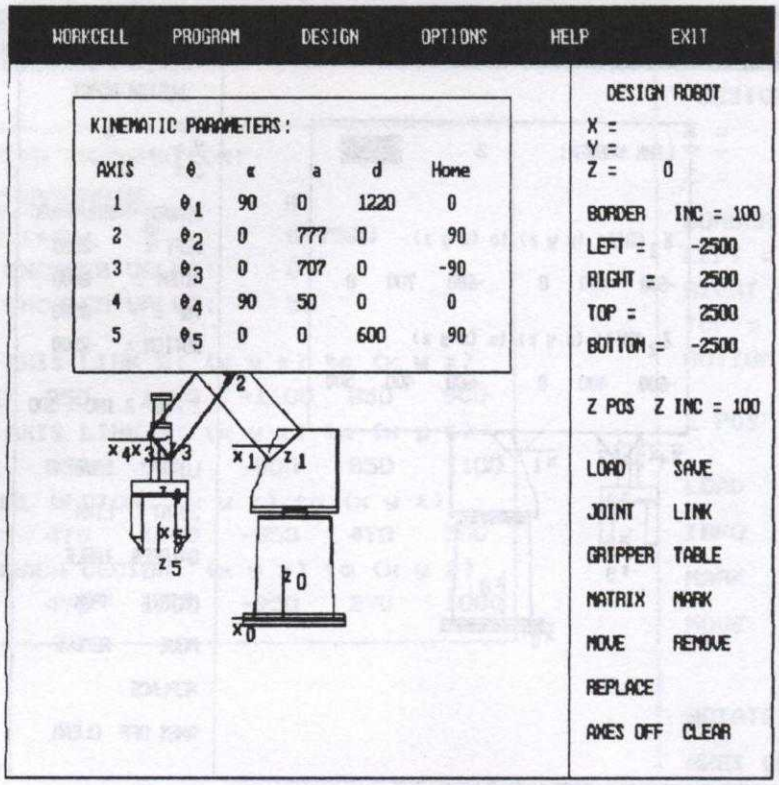

Fig. 6. A sample TABLE command.

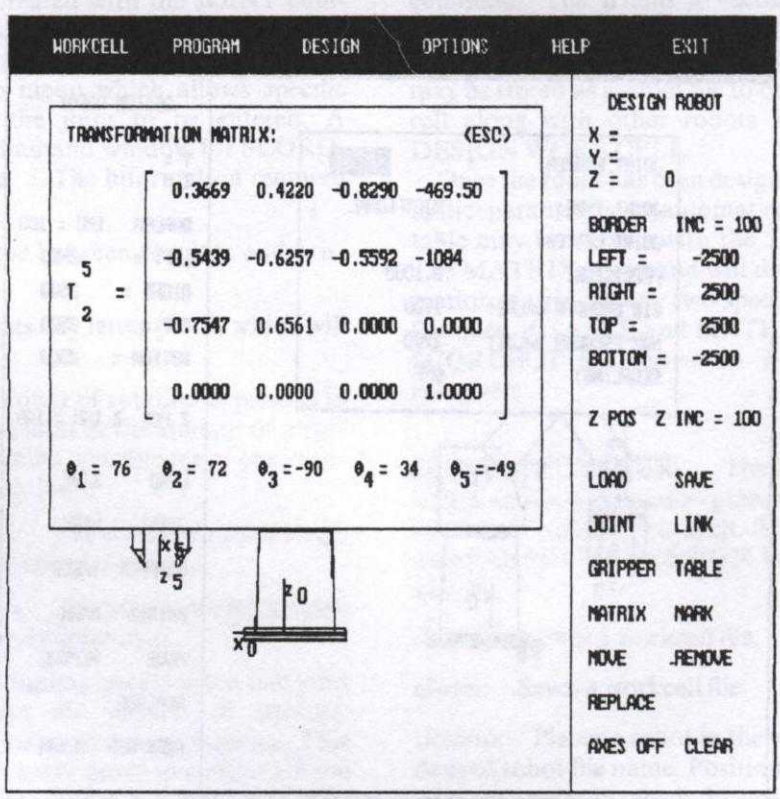

Fig. 7. A sample MATRIX command.

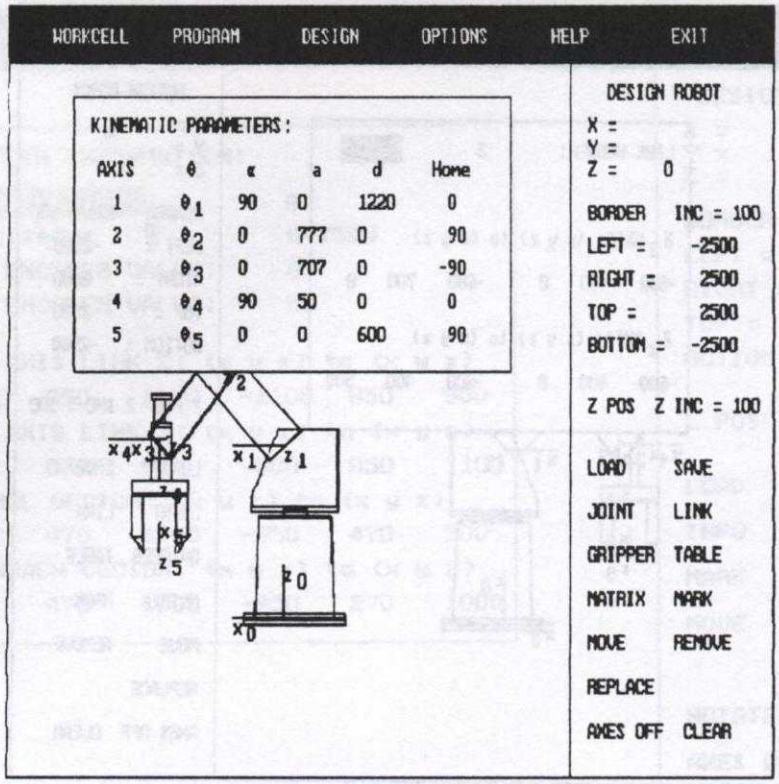

Fig. 6. A sample TABLE command.

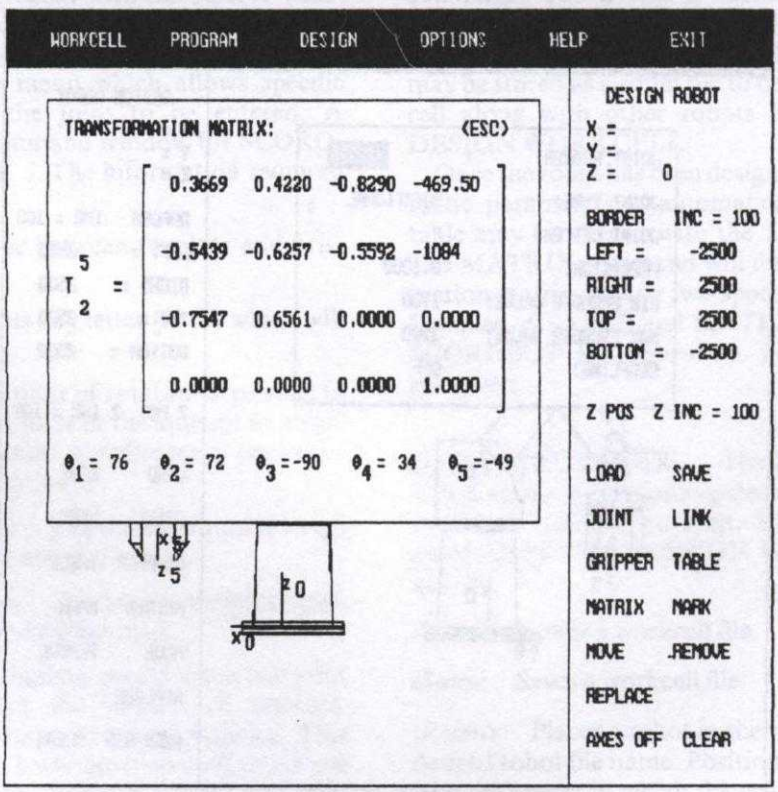

Fig. 7. A sample MATRIX command.

‹*Object*›: Places an object in the workcell. Select the desired object file name. Position the mouse's pointer to the point in which the lowest, left-most, front-most corner of the object is to be placed. Press the left mouse button.

‹*Mark*›: Displays an individual robot or object each time it is selected. Robot or object may now be edited.

‹*Replace*›: Prompts the user for a new robot or object file to replace the one marked with MARK. Place the robot or object in the same manner as the ROBOT or OBJECT command.

‹*Move*›: Prompts for an x, y and z displacement to move a marked robot or object. Displacement is entered numerically using global coordinates.

‹*Remove*›: Removes a marked robot or object.

‹*Rotate X*›: Mouse left button rotates entire workcell or a single marked robot or object about the positive x axis. Mouse right button rotates entire workcell or a single marked robot or object about the negative x axis.

‹*Rotate Y*›: Mouse left button rotates entire workcell or a single marked robot or object about the positive y axis. Mouse right button rotates entire workcell or a single marked robot or object about the negative y axis.

‹*Rotate Z*›: Mouse left button rotates entire workcell or a single marked robot or object about the positive z axis. Mouse right button rotates entire workcell or a single marked robot or object about the negative z axis.

‹*Clear*›: *Erases workcell.*

OPTION module

The OPTIONS main menu choice allows the user to set special program parameters including sound, menu background colors, mouse sensitivity, animation speed and file directories. OPTIONS is also used to enable and disable serial communication with an actual robot controller.

HELP module

The HELP main menu command displays a context-sensitive help window which briefly explains all ROBOTSIM commands.

A ROBOT DESIGN EXAMPLE

The four-axis Rhino SCARA robot is one of the robots which was modeled using ROBOTSIM. The first step in creating the Rhino SCARA was to obtain a set of measurements of the dimensions of each link. These link dimensions are then used to draw a wire-frame picture of the robot's front and side views on graph paper. A global coordinate frame is assigned to the graph paper drawing. Excessive details which make the drawing too complicated were avoided. In the case of the Rhino SCARA the links were simplified by drawing the top and bottom plates of links 1 and 2 as rectangular boxes. Both of these links were drawn with the same height even though link 2 is actually shorter. This allowed link 2 to fit within the top and bottom plates of link 1. These efforts to simplify the details did not in any way change the effect provided by the simulation. The global x, y and z coordinates of each endpoint of each line were then identified using the graph paper drawing. The selection of the scaling for the graph paper drawing was critical. The scaling requires that all endpoints be identified as integers. A scaling of 5 grid lines on the graph paper to 1 cm of robot dimension was selected to accommodate this requirement.

The graph paper drawing was then used to create each individual link as a link file using DESIGN LINK. The global x, y and z coordinate values from the drawing may be used to design each robot link as a collection of lines. A different drawing color was used for each link. The position of each link within the world coordinate frame was of extreme importance. Each link had to be drawn in its home position exactly as it appeared in the

Table 1. Link 0 box coordinates

BOX	X	X	Y	Y	Z	Z
Sub-base	0	150	-115	-125	75	75
	0	0	-115	-125	75	-75
	150	150	-115	-125	75	-75
	0	-150	-115	-125	-75	-75
Base	30	120	-115	-105	40	40
	30	30	-115	-105	40	-40
	120	120	-115	-105	40	-40
	30	120	-115	-105	-40	-40
Upright	75	105	-105	125	15	15
	75	75	-105	125	15	-15
	105	105	-105	125	-15	15
	75	105	-105	125	-15	-15

Table 2. Link 1 box coordinates

BOX/ LINE	X	X	Y	Y	Z	Z
Top	-35	125	75	75	-20	20
Bottom	-35	125	25	25	-20	20
Joint 0	75	105	25	75	15	15
	75	105	25	75	-15	-15
	75	75	25	75	-15	15
	105	105	25	75	-15	15
Joint 1	-15	-15	75	25	0	0

graph paper drawing. Tables 1 and 2 show the box and line coordinates that were entered to create links 0 and 1, respectively.

Just as the robot links required separate designs, the gripper links also needed separate designs. This required the design of the two fingers and the base of the hand. Each of these gripper links were designed in the exact same manner as the robot links except that all three were drawn in the same color. Table 3 shows the box and line coordinates for one of the fingers of the gripper.

The next step required the assembly of the gripper links using the DESIGN GRIPPER submenu. Each gripper link was loaded using the LINK command. The rotational axes r1 and r2 were designed to allow the fingers of the gripper to open and close properly. Each axis was identified by two endpoints using global coordinates. The normal and approach vectors were established in a similar manner. Values for the precision, minimum and maximum encoder values, and the joint letter designation were also assigned. Table 4 lists these gripper parameters for the Rhino SCARA robot.

Once the individual links and the gripper were designed, the robot was assembled using the DESIGN ROBOT submenu. The first step was to load each link file in succession, beginning with link 0. As each link was loaded, the x and z coordinate axes were defined using the D–H convention. These axes are defined by specifying the endpoints using global coordinates in the same fashion as DESIGN GRIPPER. Table 5 shows the values used to define these axes for each link.

The next step required the specification of each joint. This included the joint type, letter designation, precision, minimum and maximum encoder values, and joint coupling. The precision was determined experimentally since there was no specification sheet available. This was obtained by rotating each joint on the actual robot by 90° and comparing the two motor encoder values before and after the rotation. The precision was calculated as the difference between the two motor encoder values divided by 90°. The prismatic joint precision was calculated by taking the maximum distance that the joint can move and dividing it by the total variation in encoder units for that distance. The letter designations were obtained from the Rhino SCARA manual. The SCARA has no coupling and so the coupling value was off for each joint. Table 6 lists the information for joints 1–4.

The final step of the design was to load the SCARA gripper using the GRIPPER command. The final result was the simulated Rhino SCARA robot as shown in Fig. 8 which matched the operation of the actual robot identically. This was confirmed and tested using a simple workcell design and by interfacing directly with the Rhino Mark III controller and the actual Rhino SCARA robot.

Table 3. Gripper 'finger' link box coordinates

BOX/ LINE	X	X	Y	Y	Z	Z
Top	5	-155	75	75	-20	20
Bottom	5	-155	25	25	-20	20
Joint 1	-15	-15	75	25	0	0
Joint 2	-150	-120	25	75	15	15
	-150	-120	25	75	-15	-15
	-150	-150	25	75	-15	15
	-120	-120	25	75	-15	15

Table 4. Gripper parameters

BOX	X	X	Y	Y	Z	Z
Link 3	-130	-140	-5	140	-10	-10
	-130	-140	-5	140	10	10
	-130	-130	-5	140	-10	10
	-140	-140	-5	140	-10	10

Table 5. Link coordinate axes

BOX	X	X	Y	Y	Z	Z
Link	-125	-145	-5	-10	5	5
	-125	-145	-5	-10	-5	-5
	-125	-125	-5	-10	-5	5
	-145	-145	-5	-10	-5	5

CONCLUSION

ROBOTSIM has proven to be a valuable tool for students learning the fundamentals of robot motion and programming. It is especially helpful to students for visualizing fundamental concepts of robot kinematics, robot modeling and workcell design. It also provides the equivalent of hands-on experience found in an actual robotics lab.

Table 6. Joint parameters

	JOINT			
PARAMETER	1	2	3	4
Type	Rot	Rot	Trans	Rot
Letter	E	D	C	B
Precision	0.1175	0.2233	0.1200	0.2233
Min Encodr	-1000	-550	-350	-700
Max Encodr	1000	550	350	700
Coupling	OFF	OFF	OFF	OFF

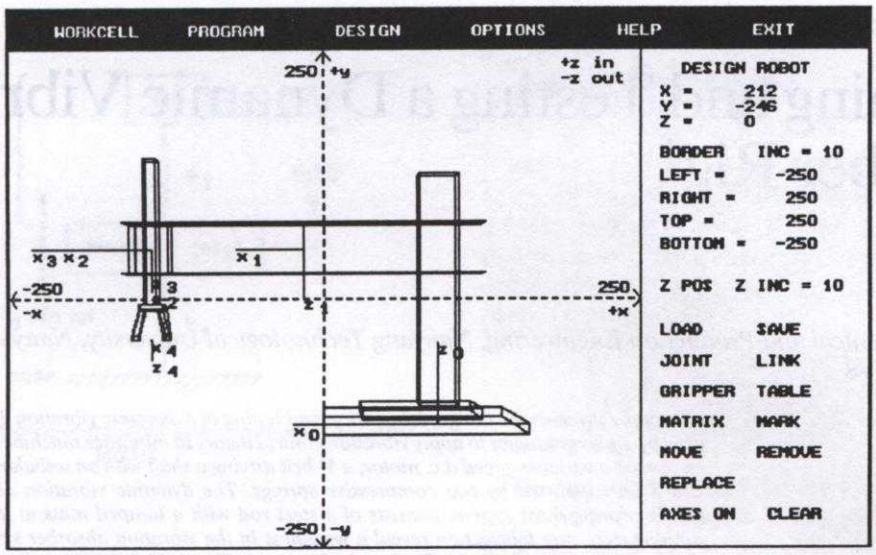

Fig. 8. SCARA robot using ROBOTSIM.

The ROBOTSIM package has been successfully used as a tool for robot modeling, programming and workcell design by students working in the area of robotics. Those interested in obtaining a copy of the ROBOTSIM package should send their request to the author.

Acknowledgement—The author would like to thank Mr. J. Sheehan for his contribution to this project.

REFERENCES

1. P. J. McKerrow, *Fundamentals of Robotics*, Addison Wesley, Reading, MA (1991).
2. T. Yoshikawa, *Foundations of Robotics*, MIT Press, Cambridge, MA (1990).
3. R. J. Schilling, *Fundamentals of Robotics*, Prentice Hall, Englewood Cliffs, NJ (1990).
4. E. I. Rivin, *Mechanical Design of Robots*, McGraw-Hill, New York (1987).
5. K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensor, Vision, and Intelligence*, McGraw Hill, New York (1987).
6. N. Viswanadham and Y. Narahari, *Modeling of Automated Manufacturing Systems*, Prentice Hall, Englewood Cliffs, NJ (1992).
7. A. C. Staugaard, *Robotics and AI*, Prentice Hall, Englewood Cliffs, NJ (1987).
8. C. Blume and W. Jakob, *Programming Languages for Industrial Robots*, Springer Verlag, New York (1986).
9. R. J. Schilling and R. B. White, *Robotic Manipulation: Programing and Simulation Studies with Software*, Prentice Hall, Englewood Cliffs, (1990).
10. T. Raz, Graphics robot simulator for teaching introductory robotics, *IEEE Trans. Ed.*, **32**, pp. 153–159 (1989).

Ali M. Eydgahi was born in Tehran, Iran. He received MS and Ph.D. degrees in electrical engineering from Wayne State University. From 1986 to 1992 he was at the Electrical Engineering Department of the State University of New York at New Paltz. Since 1992 he has been with the ECE department of the University of Tehran as an associate professor. Dr Eydgahi has served as a Deputy Director for Education and Technology Transfer at the Iran Telecommunication Research Center and also as a Vice President at the College of Telecommunication. His research interests are in the areas of robotics, control systems, computer-aided design and simulation, computer algebra systems and engineering education. He has published more than fifty journal and conference papers in these areas. Dr Eydgahi was the recipient of the Dow Outstanding Young Faculty Award in 1990 from the American Society of Engineering Education and the silver medal for outstanding research contribution from the International Conference on Automation held in India in 1995. He is a senior member of IEEE and SME, and a member of ASEE, Computer Society of Iran, and Iranian Society of Control and Instrumentation Engineers.