1997 Deneb International Simulation Conference, Troy, Michigan, September 29 to October 3, 1997.

Remote Graphic Programming and Monitoring Tools of the NIST RoboCrane® Controller

Nicholas Dagalakis
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Abstract

Simple programming and monitoring tools had to be developed to control the NIST RoboCrane® to accommodate crane operators. The programmed move instructions are generated by an off-line graphic animator, and the data files are transferred through the internet. The data are of neutral text format and any robot controller with the proper interpreter can understand them. The controller operation can be monitored from a remote location, and control instructions can be issued remotely. An animation feedback interface allows for the graphic monitoring of the operation of the robot, the display of position errors, and sensor outputs. Various network tools are being tested for the real time transfer of data necessary for the last two interfaces. Again, the interface data are of neutral format.

Biography

Dr. Nicholas G. Dagalakis received his Diploma in Mechanical and Electrical Engineering from the National Technical University of Athens, Greece in 1969. He received his M.S., Eng.D., and Ph.D., from the Massachusetts Institute of Technology, Mechanical Engineering Department in 1971, 1973 and 1975 respectively. Dr. Dagalakis worked as a Research Assistant and later as a Research Associate for the Mechanical Engineering Department and Electrical Engineering Department of the Massachusetts Institute of Technology. Before joining the Intelligent Systems Division (ISD) of the National Institute of Standards and Technology (NIST), Dr. Dagalakis worked as an Assistant Professor at the University of Maryland, College Park.

Introduction

The experimental prototype of a cable driven RoboCrane® was developed at NIST in the late 1980's [1]. Because crane operators are not expected to program crane controllers to perform complex operations, it became necessary to develop an off-line graphic programming capability for the RoboCrane® controller. Many applications planned for the RoboCrane® involve operations in hazardous environments, creating a need for development of a remote monitor and control capability. Fig. 1 shows a block diagram of the controller and its interfaces to the remote graphic programmer, monitor-controller, and animation feedback. This paper describes the techniques used to model and animate the operation of the RoboCrane® and to develop the remote graphic programmer. It also describes animation feedback capabilities mentioned above using commercially available software packages.

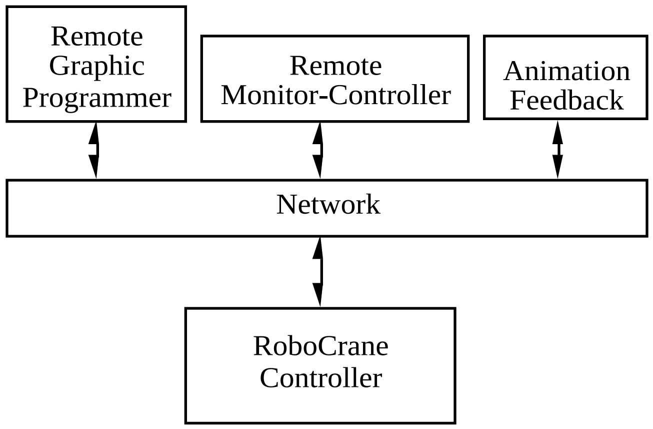


Fig. 1 Block diagram of controller interfaces

As work on these subjects progressed, it became obvious that the capabilities that were being developed had broader applications that could benefit other advance projects like the “Hexapod Machines Simulation and Characterization” [2] and the “Machine Tool Performance Models and Data Repository” [3]. The availability of network communications tools made the network a preferred medium to transfer data among the off-line programming computer, the remote monitor-controller, and the RoboCrane® controller. Experimental prototypes of interfaces to the off-line graphic programmer and the remote monitor-controller have been developed and are being tested on various demonstration problems. All these interfaces are in a neutral text format and can be understood by any controller with the proper interpreter.

Graphic Animation Model

Fig. 2 shows a TGRIP[1] model of the RoboCrane® (the role of the transparent platform will be explained later), where the supporting legs of the reference platform have been removed, to allow a better view of the mechanism.

The RoboCrane® consists of a reference frame platform and a moving platform [1]. The moving platform is suspended from the reference platform by six cables, two at each vertex of an equilateral triangle. The wires form a Stewart mechanism [4], which provides the moving platform with six degrees of freedom as long as all six cables are in tension. By controlling the lengths of these cables, the operator can control the position and orientation of the moving platform. Various types of tools can be mounted on the moving platform. These can then be used for various applications like heavy load maneuverability, construction, clean up of nuclear and toxic waste, grinding, and welding. The main advantages of this robot, as compared to a serial manipulator of the same payload, are higher payload to weight ratio, larger workspace, and higher stiffness. The main disadvantage is lower dexterity.

When the animation model of the RoboCrane® was being developed, no commercial software package could support its kinematics. Only serial types of robot mechanisms could be supported by the available software packages. The alternatives were 1) develop our own animation software or 2) make creative use of the existing ones, which is what we chose to do.

The Deneb TGRIP[1] software was used for all the animations described in this paper. This software allows the representation and control of rigid bodies moving in three dimensional space with an option called “simple inverse kinematics.” This gave the idea of breaking the mechanism into the moving platform (tool mounting plate) and several serial manipulators. The moving platform is commanded to move according to the requirements of the application, while the serial manipulators are forced to follow certain points (tag frames) of the moving platform. In the case of the RoboCrane®, these points are the suspension points of the moving platform. An alternative is to use datum lines to represent the serial manipulators.

The technique worked well and has been used for the animation of various experimental robotic devices including the Hexapod machine tool [5]. The technique is relatively simple and allows the animation of complex experimental robotic devices (combinations of serial and parallel mechanisms) with the minimum amount of time and effort.

During motion simulation, the TGRIP[1] animator generates position and orientation information for the moving platform at each time step. The information consists of the x, y, and z Cartesian coordinates and orientation of the coordinate frame during the CAD

[1] Certain commercial products, are identified in this paper to specify experimental procedures adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

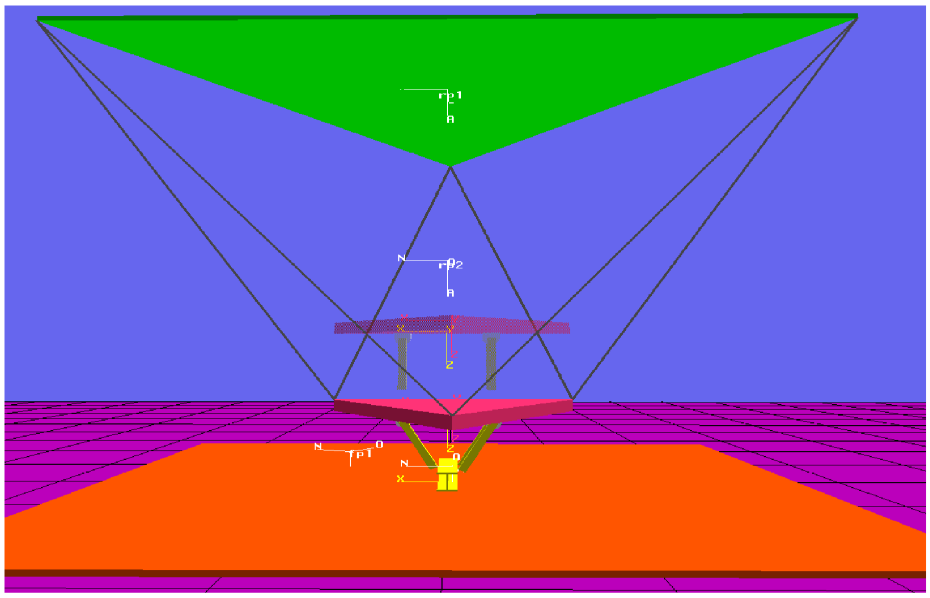


Fig. 2 TGRIP Model of the RoboCrane.
The lowest platform shown is called the "true model" while the transparent platform is called the "ghost."

creation of the moving platform with respect to the workcell base frame. In our case, this frame is located at the centroid of the suspension plane of the moving platform. This plane is defined by the suspension points of the platform. If the position and orientation of the reference platform coordinate frame with respect to the base frame is known, then the robot controller can determine the commanded position and orientation of the moving platform with respect to its own base frame.

Animation Graphic Programming

This capability allows to program a controller from the graphical animation of a desired sequence of operations.

Besides the simulation graphic sequences generated by an animator, the animator can also be programmed to generate a basic time history of robot motions and tool actions in neutral text format. A robot controller may then connect to this animation graphic programmer and read the time sequence of instructions. With the proper interpreter, the controller should be able to understand these instructions and command the robot to repeat the motions and actions simulated by the animator.

This low level interface would allow simple inexpensive controllers to control complex workcells, since there will be no need for the controller to be equipped with trajectory planning software.

The transfer of information between the controller and the graphic programmer could take place by a direct serial or parallel connection, or a network connection. In the case of our RoboCrane® controller, the network connection was chosen. A network transfer of data offers the advantage that a single animator graphic programmer can program a large number of controllers, which makes optimal use of resources. A remote graphic programming capability frees the crane operator or plant floor operator from typing instructions of long computer programs in a noisy environment, which is prone to typing and other mistakes.

The objective of our effort until now has been to develop a neutral interface to transfer data from the graphic programmer to the controller. The programming of the animator, itself, was left to the individual programmer. Several commercial robot graphic animation software packages are available. Each one of them uses different programming techniques and instructions.

In the case of long trajectories, the graphic programming interface file, with low-level time sequences of instructions, could become large. In those cases, it might be better to save high-level motion instructions, which can be in a neutral text format that is easily recognized by the controller. For this high-level interface to work, the controller must use the same names for the trajectory points as those used by the animator. Unfortunately, all available robot controllers use different programming instructions. For this interface to become possible, a standard set of instructions must be adopted.

Several RoboCrane® applications were programmed using the remote graphic programming technique. One of the most complex was an arc welding application. Fig. 3 shows the RoboCrane® welding workcell.

The moving platform carries two necessary tools for this application, a grinder and a welding wand. The grinder cleans the surfaces before welding. It retracts when not in use, and it is lowered below the bottom surface of the platform when it is time to grind. The controller can control the power and position of the grinder, although manual control is possible too. Similarly, the welding wand can be retracted when not in use and lowered when welding must take place.

The graphic programmer generates two files. The first file specifies how many robots are active in the workcell and how many tools they carry. The second file contains the necessary control data to carry out the welding operation. Fig. 4 shows a portion of that file.

The first column represents time. The next three columns provide the absolute position of the moving platform (mounting tool plate) centroid with respect to the work cell base frame. The next three columns provide the angular rotations of the platform axes with respect to the base frame. And the last four columns represent binary information. The first bit specifies whether the tools are under manual or automatic control. The second indicates whether the welding wand should be lowered or retracted. The third controls the welder power. The last bit controls the position and power of the grinder. At this particular segment of the operation, the data indicates that the controller must have control over the tools, and the grinder must be at its lower position grinding, as Fig. 3 shows. As soon

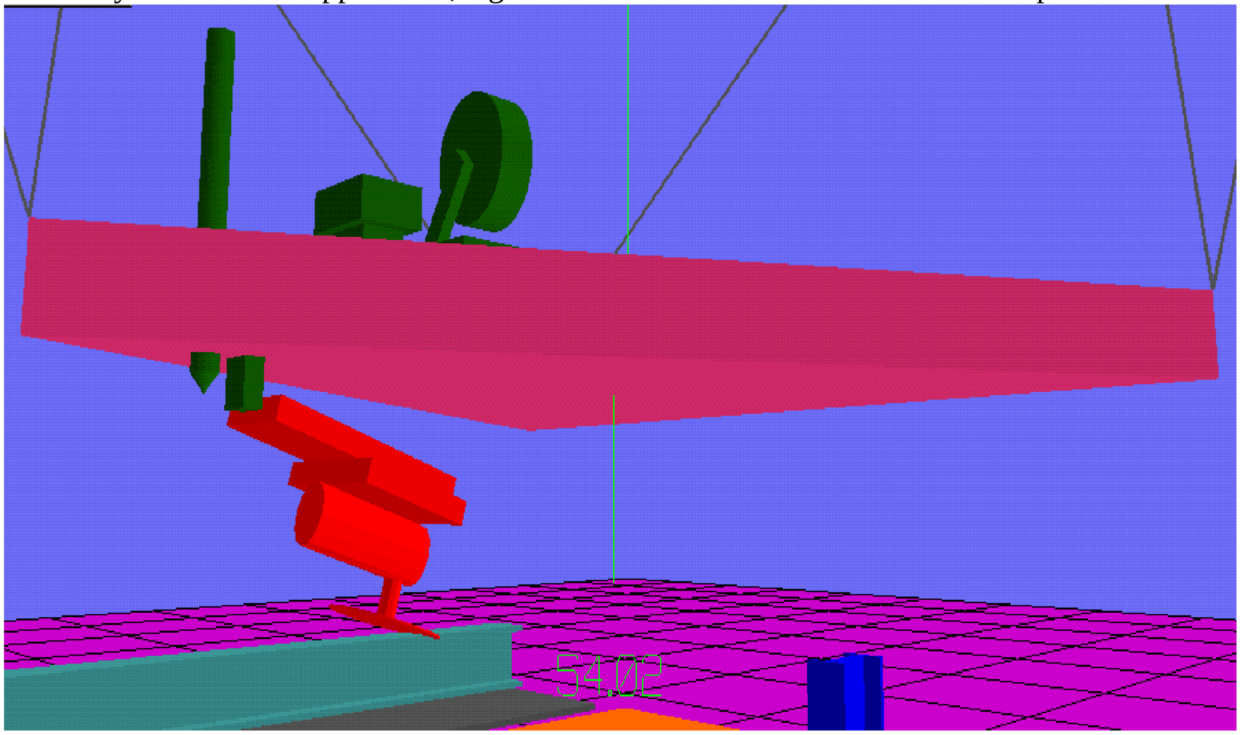


Fig. 3 TGRIP Model of the RoboCrane Welding Workcell.

The grinder is being controlled followed by welder extension from the platform and welding of the beam

Time	X	Y	Z	Φ_z	Φ_y	Φ_x	$b_1b_2b_3b_4$
.	.	.	.	.	.	.	. . . .
50.150	0.000	27.756	54.052	0.000	0.000	180.000	1 0 0 1
50.320	0.000	28.011	54.051	0.000	0.000	180.000	1 0 0 1
50.490	0.000	28.266	54.050	0.000	0.000	180.000	1 0 0 1
50.660	0.000	28.521	54.048	0.000	0.000	180.000	1 0 0 1
50.830	0.000	28.776	54.047	0.000	0.000	180.000	1 0 0 1
51.000	0.000	29.031	54.046	0.000	0.000	180.000	1 0 0 1
51.170	0.000	29.286	54.045	0.000	0.000	180.000	1 0 0 1
51.340	0.000	29.541	54.043	0.000	0.000	180.000	1 0 0 1
51.510	0.000	29.796	54.042	0.000	0.000	180.000	1 0 0 1

Fig. 4 Sample of control data

as a tool is activated, its color turns red in the animation scene and remains red until it is deactivated. Programs of sequential operations can be accommodated by appending the graphic programming data to the data file of the previous program. A pop-up window allows the operator to select, to save or not to save, append or abort the file.

Because the parts used for welding are of construction quality, there is significant variation in their dimensions. No seam tracking sensor has been installed yet, so this necessitates the use of manual teach control to supplement the off-line programming and align the graphic model of the part to that of the RoboCrane® workspace. Under manual control, the moving platform is moved so that the welding wand touches a few key points of the parts. The corresponding graphic model points (tags) are then moved to assume the same coordinates before graphic programming commences.

Fig. 5 shows a flow chart of the controller interface. The connection to the graphic programmer is activated by a button on the front panel of the controller (see Fig. 6). When a connection is established, the button turns red and remains red until the complete transfer of the data.

First, the file with the control data must be read. The control data specify the number of active robotic devices and the method used to specify the translation and rotation of their tool plate for example a sequence of roll, pitch, yaw rotations. The number of tools each one carries must also be specified at this point. This information defines the type of command data (format) that the graphic programmer generated.

The controller interface remembers the number of commands it received from the graphic programmer. Every time the controller interface is called, it discards the old data and works with only the new data. The new data are read and their format is thoroughly checked. If any discrepancy is found, the operation is aborted. The frequency of occurrence depends on the communications hardware and the network transfer protocol used. The User Datagram Protocol (UDP) should only be used when the underlying network is inherently reliable. The UDP is a network communications protocol that does not provide date packet sequencing and provides an optional check sum.

Once the data have passed all these checks, they are ready to be used. One line of data is read at a time. The moving platform move commands are separated from the binary tool control data. The move commands are checked for limit violations and then sent to the controller. The tool commands are sent to the binary interface and Light Emitting Diode (LED) indicators on the control panel (see Fig. 6).

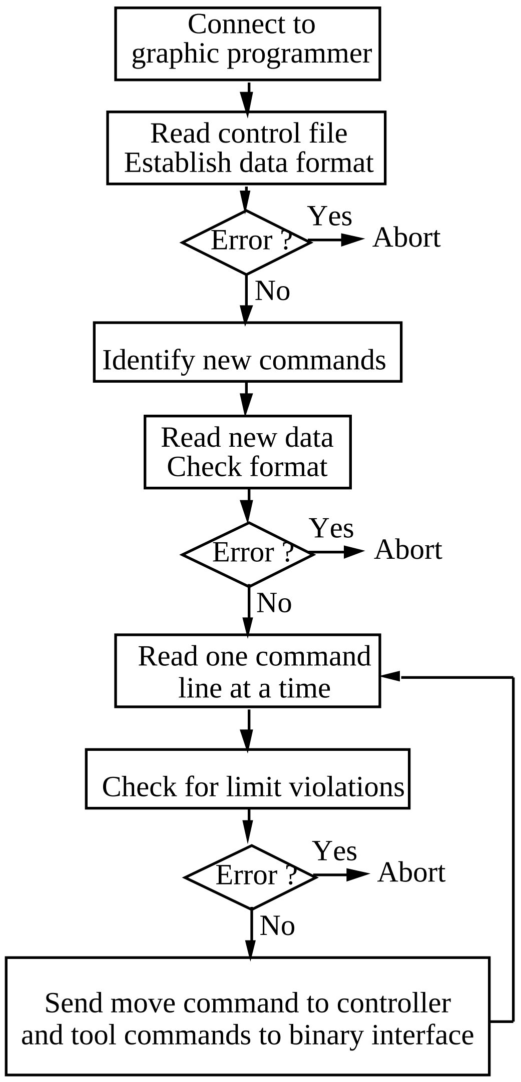

Connect to graphic programmer
Read control file Establish data format
Error ?
Yes
Abort
No
Identify new commands
Read new data Check format
Error ?
Yes
Abort
No
Read one command line at a time
Check for limit violations
Error ?
Yes
Abort
No
Send move command to controller and tool commands to binary interface

Fig. 5 Graphic programming controller interface

Animation Feedback

Animation feedback is an animation graphics capability with two solid models of anything that moves in a workcell. One model, "ghost", is transparent; the other, "true model", is not. The animation feedback moves the "ghost" model based on its commanded (desired) position and orientation, while the "true model" is moved based on its measured (true) position and orientation. Time delays in the execution of the move commands and calibration errors cause a difference in the position and orientation of the two models.

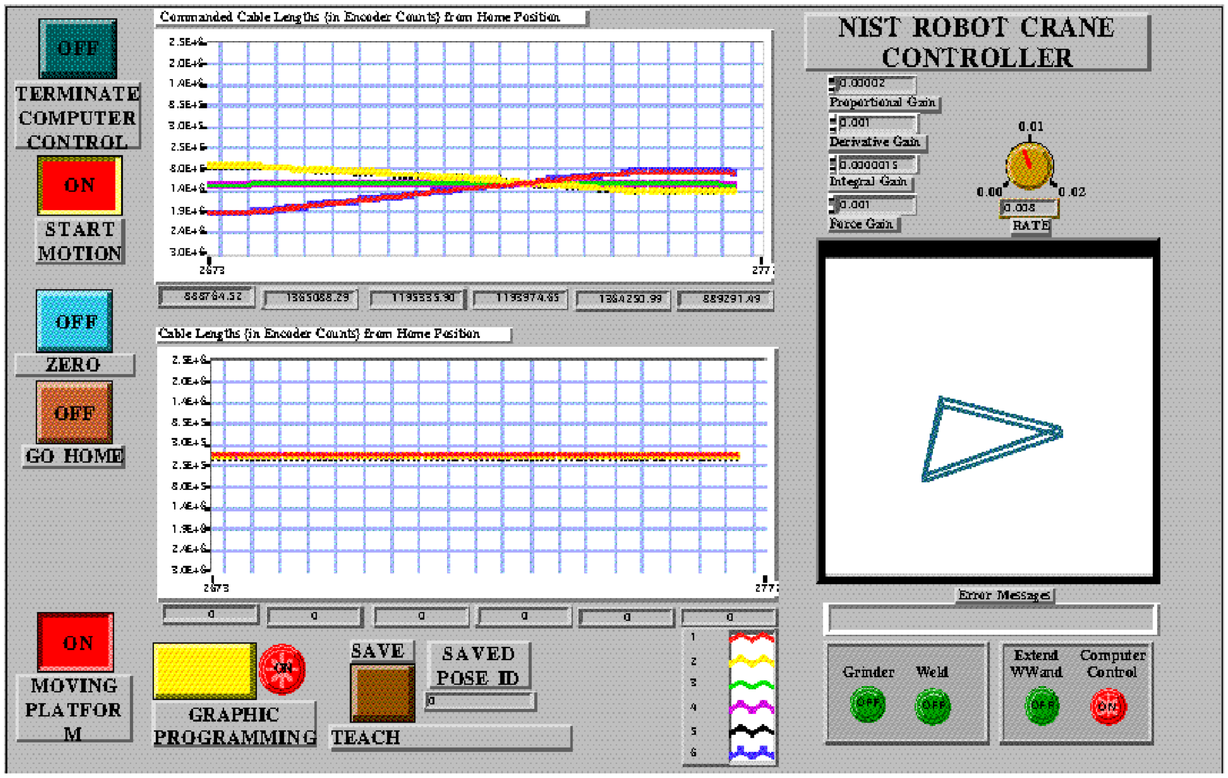
Fig. 2 shows the ghost transparent figure of the RoboCrane® moving platform and the true model. In this case, the ghost delivered an "I" beam and it is moving away while the true model has not completed the task yet. Fig. 7 shows a similar case involving a machine tool. The spindle of this machine tool is commanded to follow the path tracked by the ghost. In reality, the true model follows a different path which results in machining errors.

There are two different applications of this capability. One is the simulation of faulty operation of a device and the animation of the results this has on the operation. Another is the real-time animation display of the device operation where the true model is driven by the measured position and orientation of the device. We have experimented with a network socket interface between the controller of the RoboCrane® and its TGRIP[1] graphic model to transfer animation feedback position and orientation information.

The animation feedback interface is useful for cases where the robot operator has no direct line of sight view of the workcell and video cameras are not available or too slow. The operator can view the differences between the desired and actual position and orientation of all moving parts, and pan, tilt, rotate, or zoom to obtain a better view of these differences. The differences (errors) can be multiplied with a scale factor in order to exaggerate the visualization effect. This way, a small defect can be noticed and corrected before a catastrophic failure occurs. In the case of a simulated faulty operation, a better insight into the operation of the device can be obtained.

Two separate processes generate the twin images of the animation feedback, ghost and true model. These processes communicate with each other through socket ports. When one process generates a command, for example, "move the true model to a certain location," the other process is notified to execute the corresponding command for the twin device model. In this case, the ghost is commanded to move to the corresponding desired location. No further action is allowed until both devices have completed the command execution.

position even after the tools have moved away. The color of each line is green if the



888764.52 | 1385088.29 | 1195335.90 | 1195974.65 | 1384290.99 | 889291.49

0 | 0 | 0 | 0 | 0 | 0

Proportional Gain 0.00002
Derivative Gain 0.001
Integral Gain 0.0000015
Force Gain 0.001
RATE 0.008
SAVED POSE ID 0

Fig. 6 Control panel

The display of error vectors is an extension of the animation feedback twin images idea. These vectors are attached to specific parts of devices and can display position errors or orientation errors. They are drawn in such a way that they follow the movements of the part to which they are assigned. The vectors' lengths are scaled representations of the magnitude of the errors they represent. Their orientation can be used to display the directional property of the error. Their color can be used to represent another property of the error.

Fig. 7 is an example of the use of this technique for the display of machining position errors. The straight lines, resembling whiskers, connect corresponding points between the desired and the true (achieved) locations of the cutting path. As the ghost and the true models of the spindle are moving, the error vectors are drawn as straight lines, which connect the centers of the two cutting tools and remain in that

tool cuts less material than specified, and red if it cuts too much material. Since machining errors are very small and difficult to see, they must be multiplied with a large scale factor to make them more visible. In the case of Fig. 7, the scale factor selected by the operator was 10,000.

Another interesting application of these animation vectors is the display of sensor outputs. For example, these could be the outputs of temperature, or accelerometer sensors, mounted on the spindle and/or actuators of a machine tool or end effector and/or links of a robot. Since these attached vectors follow the moving parts, they can provide a very vivid view of the condition of the device.

Drawing error vectors, attached to graphic models of three degrees of freedom (3-D) animations, is not trivial. Most commercially available software packages do not provide such a capability. To generate these vectors in TGRIP[1], fictitious parts had to be imported into the workcell. There were as many parts as the desired colors of the error vectors. Before the drawing operation, the part of the desired color would be activated (opened). The two ends of the vector are marked by coordinate frames (tags) which are attached to the desired part.

One way to determine whether the tool is excessively penetrating into the part is to transform the homogeneous coordinates point D vector $\underline{V}_d$ from base frame coordinates to a coordinate system attached to A, as shown in Fig. 8. If $\underline{\underline{T}}_{BA}$ is the homogeneous transformation from the base frame to a frame attached to A, then

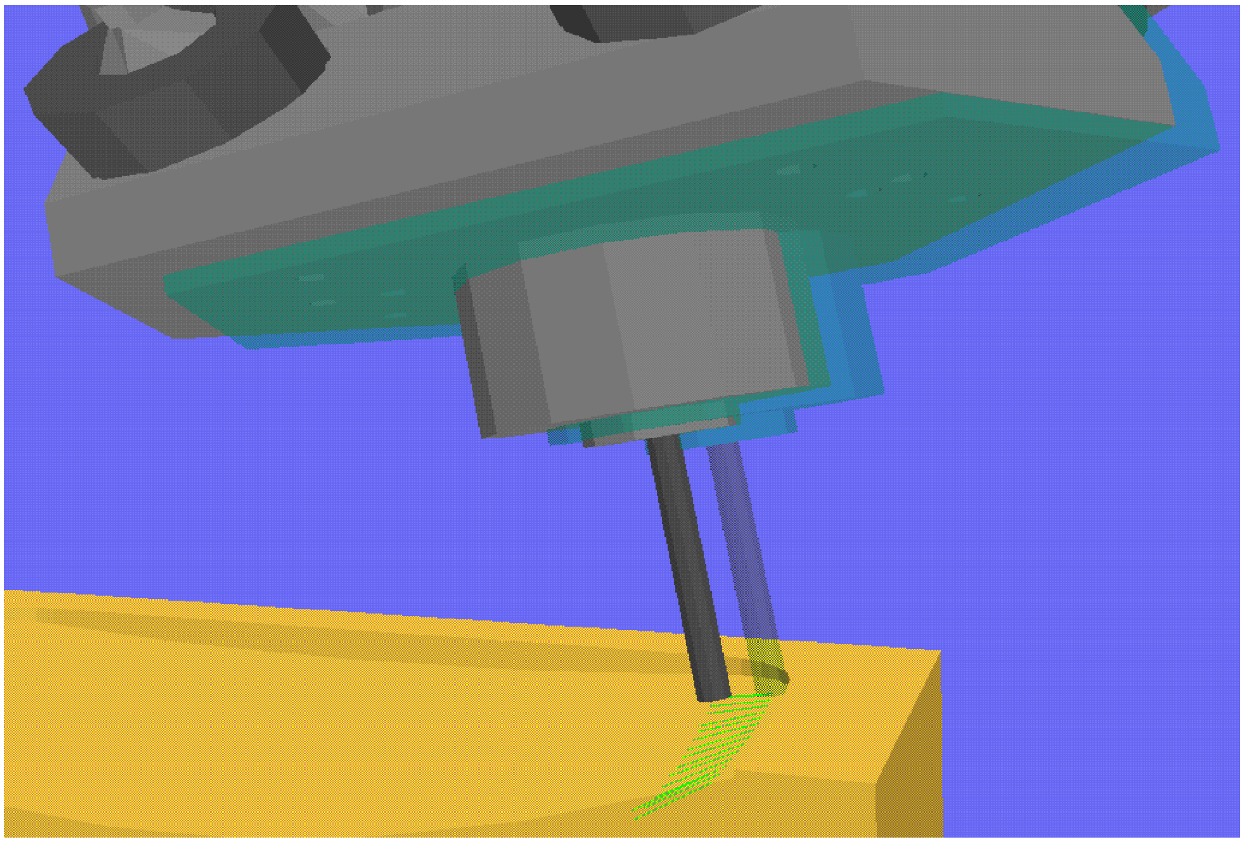


Fig. 7 Hexapod spindle gost image generates error vectors

$$\underline{V}_{da} = \underline{\underline{T}}_{BA}\,\underline{V}_d \quad (3)$$

A positive value of the coordinate of $\underline{V}_{da}$ along a certain cutting direction indicates that the tool cuts less material than it is supposed to cut. A negative value indicates the opposite. The operator specifies, at the beginning of the simulation, the direction along which material is being removed. In the case of Fig. 7, it is an end cutting operation, and the Z axis coordinate of $\underline{V}_{da}$ determines the color of the error vector.

Fig. 8 shows the vector positions of the ghost and true models cutting tools with respect to the base frame. D represents the desired position of the tool tip, given by $\underline{V}_d$, and A the achieved or true position given by $\underline{V}_a$. S is the scaled error vector position given by $\underline{V}_s$. The error vector is calculated as:

$$\underline{V}_e = \underline{V}_a - \underline{V}_d \quad (1)$$

If SF is the desired scale factor then

$$\underline{V}_s = SF * \underline{V}_e + \underline{V}_d \quad (2)$$

The true model of the tool must be translated by the offset vector $\underline{AS}$.

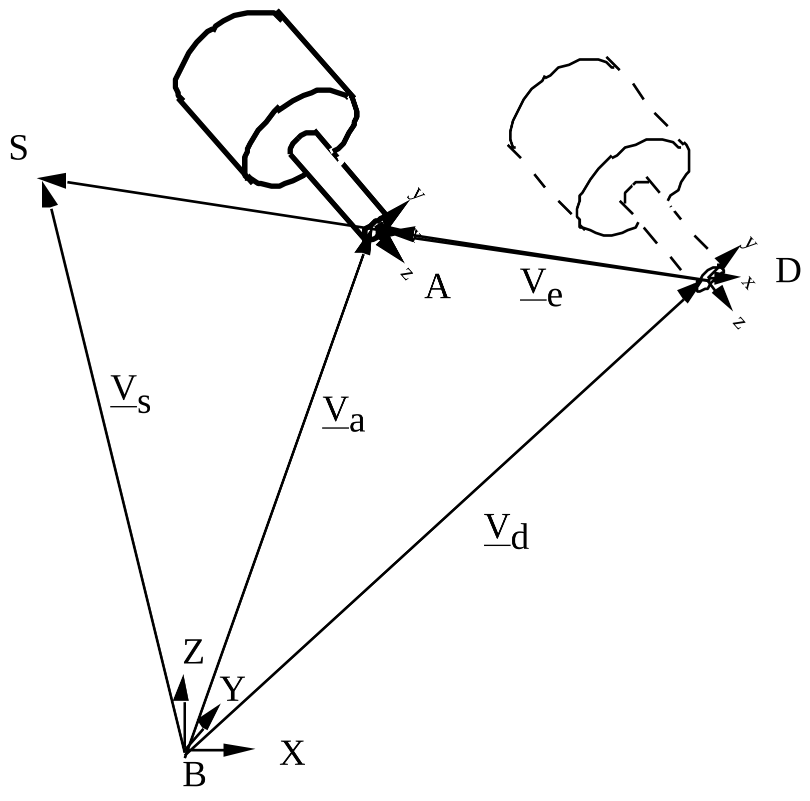


Fig. 8 Vector representation of errors

Conclusions

Two capabilities of the RoboCrane® controller have been described. The remote graphic programmer can reduce the cost of programming and remove it from the error prone plant floor environment. The network interface allows for the optimum use of an expensive resource, and the animation permits inspection prior to testing. The animation feedback simplifies the study of faulty behavior of robotic and machine tool devices. Its use as a real time monitoring tool could reveal the presence of defects and prevent catastrophic failures. The vector visualization associated with animation feedback could be very useful for the display of errors and sensor output signals.

Acknowledgments

The author would like to thank Joe Falco of NIST/ISD for preparing portions of some of the Computer Aided Design (CAD) models in figures used in this paper, laying out the machining paths of the Hexapod, and help with the Graphic Simulation Language (GSL) programming questions. I would also like to thank the late Nicholas Tarnoff who coined the term “ghost” for the transparent animation feedback figure. May God grant him peace. Special thanks go to Deborah Gilbert, Bruce Bradberry, and Carlos Perozo of Deneb Robotics, Inc., for their help to solve some of the problems associated with the use of TGRIP[1] to do the work described in the paper.

Partial support for this work was provided by the National Advanced Manufacturing Testbed (NAMT). This is a testbed to demonstrate how machines, software, and people can efficiently and effectively be networked together to improve productivity and foster innovation at all levels of a manufacturing enterprise.

References

1. “The NIST ROBOCRANE®,” J. Albus, R. Bostelman, N. Dagalakis; Journal of Robotic Systems, 10(5), 709-724, 1993.

2. “Characterization Remote Access and Simulation of Hexapod Machines,” Toward 21st Century Information Based Manufacturing, NIST Special Publication 913, April 1997.

3. “Machine Tool Performance Models and Machine Data Repository,” Toward 21st Century Information Based Manufacturing, NIST Special Publication 913, April 1997.

4. “A Platform with Six Degrees of Freedom,” D. Stewart, Proc. of the Inst. of Mech. Eng., Vol. 180, Part I, No. 15, pp. 371-386, 1965-11966.

5. “Virtual Manufacturing Tools for Collaborative Exploration of Hexapod Machine Capabilities and Applications,” J. Falco, Deneb International Simulation Conference, Sep. 1997.