Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

A METHODOLOGY FOR DEVELOPING ROBOTIC WORKCELL SIMULATION MODELS

Frank S. Cheng

Department of Industrial and Engineering Technology
Central Michigan University
Mount Pleasant, MI 48859, U.S.A.

ABSTRACT

Robotic workcell simulation is a modeling-based problem solving approach developed for the design, analysis, and offline programming of robotic workcells. Current industrial practices show that commercial robotic simulation software packages are able to provide designers with an interactive and virtual environment in which credible solutions for robotic workcell designs can be obtained. However, conducting robotic workcell simulation studies via robotic simulation packages require designers to carry out complex processes of modeling, programming, and analysis, which often results in technical challenges and difficulties. In this paper, a methodology for developing robotic workcell simulation models via Deneb IGRIP technology is introduced. The development of the method is based on successful applications of Deneb's IGRIP robotic simulation software in designing real robotic workcells.

1 INTRODUCTION

Robotic workcells are important elements in automated manufacturing systems for delivering required manufacturing materials and operations with industrial robots and associated peripheral devices. Rapid design and deployment of a robotic workcell require the successful applications of concepts, tools, and methods for fast product design, manufacturing process planning, and plant floor/cell control support. An important technology for achieving this goal is robotic workcell simulation.

Robotic workcell simulation is a modeling-based problem solving approach that aims to sufficiently produce credible solutions for robotic system design. Figure 1 shows a simple robotic workcell simulation model developed using Deneb's IGRIP robotic simulation software. The current practices of robotic workcell design have proven that successfully implementing robotic simulation brings the designers many benefits (Rudnick 1997, Knasinski 1997, Craig 1997, and Rooks 1997). First,

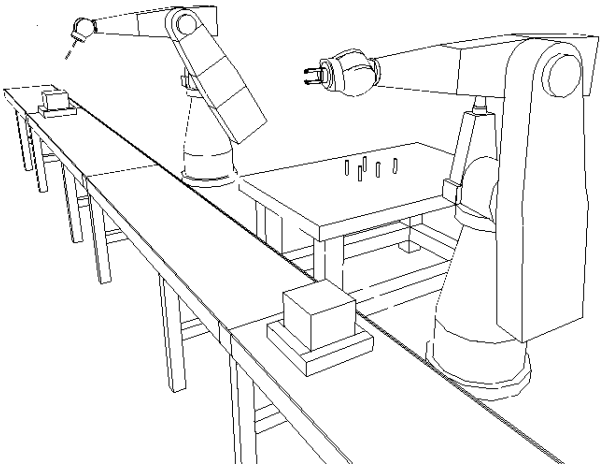

Figure 1: An IGRIP Robotic Workcell Simulation Model

designing a robotic workcell via robotic simulation eliminates the guesswork from concepts and unrealistic expectations based on technical equipment specifications.

Designers can offer optimum solutions to designs via having evaluated alternatives. Second, as modifications are made to a workcell design, the process of incorporating modifications into the corresponding workcell simulation models is mush easier and faster compared to making changes to a real workcell. Finally, robotic simulation packages bring designers a safe design environment. Whether designing a new workcell, optimizing its performance, or making modifications to an operational workcell, developing and testing required programs can be safely carried out.

Models in robotic workcell simulation are principles for studying the behavior of the actual workcell devices over time. These models may be geometric objects, mathematical equations and relations, or graphical representations. Designers usually use the commercial robotic simulation software packages to build simulation models. The simulation design tasks require designers to constitute specific methods and procedures via selecting

and executing the functions provided by the simulation packages with appropriate design data. This process often involves both inductive and deductive reasoning and requires multifaceted knowledge in diverse disciplines such as computer-aided design (CAD), machine design, and robotics (Chen and Cheng 2000). Due to the complex processes and multifaceted knowledge requirements, the robotic simulation designers often face significant theoretical and technical challenges in understanding and applying the current robotic simulation technology. To deal with this challenge, this paper introduces a methodology for developing required robotic workcell simulation models via Deneb IGRIP robotic simulation technology.

2 IGRIP OVERVIEW

The Interactive Graphics Robot Instruction Program (IGRIP) software, a product of Deneb Robotics, Inc., is a user-friendly computer based robotic workcell simulation package for robotic workcell layout design, simulation and offline programming. IGRIP is divided into three primary systems: the IGRIP Menu System, Graphic Simulation Language (GSL), and Command Line Interpreter (CLI).

The IGRIP Graphical User Interface (GUI) provides a menu driven, point and click approach to robotic simulation. A menu option on GUI is referred to as a Context. The commonly used Contexts for developing simulation models include CAD, Device, Layout, Motion, and Program. Selecting a Context will cause a pull-down menu to appear. Usually each pull-down menu displays several choices that are referred to as Pages of a Context. A Page provides access to a set of related IGRIP operations. A Title Bar on a Page serves to group and define IGRIP's Action Buttons that are displayed immediately below it. The selection of an Action Button is the basic method for performing an IGRIP's function.

The IGRIP Graphic Simulation Language (GSL) is a procedural language used to control the behavior of simulation models. GSL incorporates conventions commonly used in high-level computer languages with specific enhancements for model's motion and simulation environment inquiries.

The Command Line Interpreter (CLI) is a powerful communication, command, and control system for accessing and operating IGRIP system. It is accessible from both inside and outside the IGRIP menu system.

3 CREATING PART MODELS

A Part model in IGRIP is a low-level, named geometric entity. The IGRIP CAD Context provides operations for creating three-dimensional visual representations of Part models. The geometry of a Part model can be created in or imported into the IGRIP CAD Context via basic geometric elements such as vertices, lines, polygons, edges, surfaces, etc. The IGRIP CAD Context has a world Cartesian coordinate system that works as the common reference point for dimension measurements of Part models. The position (i.e., location and position) of the CAD world coordinate system cannot be changed in the CAD Context. When a Part model is created in or imported into the CAD Context, a base Cartesian coordinate system is attached to the Part model. The base coordinate system of a Part model is always superposed on the world coordinate system of the CAD Context as the Part model is first created in or imported into the CAD Context.

A Part model can be manipulated via translation and rotation operations in the CAD Context. After manipulation operations are performed to a Part model, it looks on the screen of GUI as if the entire Part model (i.e., the base coordinate system and the body of the Part model) has been moved to a new position in the CAD Context. However, when the Save function is performed within the CAD Context to the Part model, the base coordinate system of the Part model always superposes on the world coordinate system of the CAD Context and the body of the Part model remains in the same position as it was in the CAD world before performing the Save function. In other words, the base coordinate system of a Part model cannot be moved with the manipulation operations (i.e., translation or rotation) in the CAD Context. Only the position of geometric body of the Part model is actually moved with manipulation operations. In order to manipulate the base coordinate system of a Part model in the CAD Context, different procedures must be performed in the CAD Context.

Besides the base coordinate system of a Part model, other auxiliary coordinate systems can also be first created at the base origin of the Part model, and then manipulated to specified positions. It is important to notice that both the base and auxiliary coordinate systems of Part models are to be used in constructing an IGRIP Device model.

4 BUILDING DEVICE MODELS

A Device model in IGRIP represents an actual workcell component such as robot, table, conveyor, part, end-effector, and so on. The simplest Device model may only consist of one Part model, representing the workcell component that is defined as a rigid body such as a workpiece, peg, pallet, table, and conveyor in the workcell. These one-part models are usually called non-robotic Device models. If an IGRIP Device model consists of multiple Part models that are connected by a series of joints, the Device model is called a manipulator model or robotic model that may have the complex geometry of motions. All industrial robots are represented as manipulator models in IGRIP. For a Device model with multiple Part models, the Part models play the roles of structural elements that compose joints of the Device

model for generating motions between Part models. Usually, an independent motion generated by a joint of a manipulator or robotic Device model defines a "degree of freedom" (DOF) of the Device model.

Creating an IGRIP Device model starts with the base Part model that is the first Part model retrieved from the IGRIP Part library and placed in the Device Context. The base coordinate system of the base Part model is used by IGRIP as the base coordinate system of the Device model. Then, the second Part model is to be selected and attached to the base Part model. As a rule of attaching Part models for constructing a Device model, an available coordinate system (i.e., base coordinate system or auxiliary coordinate system) of the selected Parent Part model on the Device model must be selected to attach to the base coordinate system of the Attached Part model.

Besides the structural construction of a Device model using Part models, the programmability and communicability of a Device model can also be defined or specified when the Device model is created. The programmability determines whether or not a Device model can be programmed via IGRIP's GSL, and the communicability specifies the capability of a Device model to send and receive signals via its input/output (I/O) ports.

4.1 Motion Transformation of Part Models

The motion transformation of a Part model on a manipulator or robotic model defines how the Part model moves with respect to the joint defined by the base coordinate system of the Part model. Usually, the motion transformation of a Part model can be defined as a sequence that includes "Set-Home," "Basic Transformation," and "Return." Among the sequence, the "Basic Transformation" defines how the Part model moves (i.e., translation or rotation) about three principal axes of its base coordinate system. For an n-joint manipulator model, the motion transformation must be assigned for each of the Part models that constitute the corresponding joint, which results in n DOF. When a motion of a Part model in a Device model is dependent on other joints or DOFs, a DOF expression may also be used in IGRIP to specify the "Basic Transformation" of the Part model. Generally, a DOF expression specified for a joint of a Device model can be as simple as a constant (i.e., 1, 2, etc.), or as complex as a mathematical program.

4.2 Tool Center Point (TCP) of Robotic Device Model

A study of the geometry of motion is essential for controlling the behavior of a manipulator model. This is achieved by establishing the mappings between the position (i.e., location and orientation) of the end of the manipulator model and the movements of the manipulator Part models. The end of a manipulator model is referred to as the "Tool Center Point (TCP)" which often includes the mounting plate offset on the wrist of the manipulator or robotic model. If a Device model such as an end effector is attached to the mounting plate, the TCP may be redefined at a point on the attached Device model.

The TCP position of a manipulator model can be represented in a number of ways. One way is to utilize the joint variables of a manipulator model, which is known as the representation of the TCP in the "joint" space of the manipulator model. In this representation, each joint variable is defined as either an angle θ for a rotational joint or a linear translation d for a prismatic joint.

The TCP position of a manipulator model can also be defined in the "world" space of the manipulator model. This representation involves the use of the base coordinate system of the manipulator model and a describing frame created at the TCP. The location of the TCP frame is then represented by location values of x, y, and z measured in the base coordinate system. The orientation of the TCP frame with respect to the base coordinate system of the manipulator model is represented by the values of yaw, pitch, and roll (Paul 1981 and Rubinovitz 1999).

4.3 Kinematics of Manipulator Model

Representing the TCP in different spaces (i.e., the world and joint spaces) is served for different purposes. The TCP representation in the "world" space is very useful when a manipulator model must interact with other Device models in the workcell. With such a "neutral" representation, tasks carried out by all Device models can be specified in the IGRIP workcell layout. However, from the perspective of IGRIP motion control and simulation, given a desired position of the TCP in the workcell, the IGRIP motion controller of the manipulator model needs to know the corresponding values of joint variables in order to position the TCP at the desired position. In this case, a "joint" space representation of the TCP is critical. In order to obtain both representations, transformations must be performed from one to the other. A transformation going from the "joint" space to the "world" space is called forward kinematics, and a transformation going from the "world" space to the "joint" space is called inverse kinematics.

Given the geometry of a manipulator model, the objective of inverse kinematics is to find the corresponding values of joint variables that will place the TCP at a desired position relative to the base coordinate system of the manipulator model within its work volume. In IGRIP, an inverse kinematics solution for a manipulator model can be obtained via different methods. These methods include generic, numeric, simple, device, and user-defined methods. Depending on the nature of the actual component being modeled, the user may choose a particular method listed above to develop the required inverse kinematics for the Device model.

The generic-inverse-kinematics method makes it possible to obtain inverse kinematics solutions for most types of manipulator model with six or fewer DOFs. The method works by breaking the general kinematics problem into generic classes of manipulator models. Typical classes of manipulator models include articulate, Cartesian, cylindrical, spherical, and so on.

The device-inverse-kinematics method allows a user to copy the assigned inverse kinematics of an existing device model for a new Device model being built. However, the following conditions must be satisfied as this method is applied:

- The orientations of the base coordinate systems for each Part model on each Device model must match exactly.
- The positive/negative directions of a rotation must be identical.
- The number of DOFs for both Device models must be the same.
- The type of each DOF (i.e., rotational or translational) must be the same for both Device models.

The simple-inverse-kinematics method works for a simple or non-robotic Device model that (1) consists of only one rigid part model and (2) requires moving its body in the workcell as a free body with at least six DOFs. This method is often used for Device models such as AGVs, carts, pallets, etc, whose body need to be moved in the workcell.

4.4 Device Motion Attributes

The motion attributes of a Device model define the motion limits for the joints of the Device model in terms of home position, speed, acceleration, and travel. The attributes are stored in the database file of the Device model and thus are permanently associated with the Device model. The attribute "home position" defines the default position and orientation of joints of the Device model, resulting in the default configuration of the model. Each Device model may have multiple numbered home positions. The "speed" and "acceleration" attributes set the maximum speed and acceleration for a joint or TCP of the Device model. The "travel" attribute sets the travel limits for each joint of the Device model.

5 POSITIONING DEVICE MODELS IN IGRIP LAYOUT

The geometry of a workcell layout model is developed using the functions defined by the Workcell Page in the IGRIP Layout Context. When a Device model is retrieved from IGRIP Device library and placed in the layout, the base coordinate system of the Device model first superposes onto the world Cartesian coordinate system of the layout Context.

A Device model in the layout Context can be translated and rotated with respect to its base coordinate system or the world coordinate system of the layout. If the translation (or rotation) performed to a Device model is absolutely to the world coordinate system of the layout, the operation represents an absolute positional or rotational measure between the base coordinate system of the Device model and the world coordinate system of the layout Context. If the translation or rotation performed to a Device model is relative to its base coordinate system, the operation represents the relative positional or rotational measure between a specific axis of the Device model and the base coordinate system of the Device model.

In the IGRIP layout Context, the base coordinate system of a Device model can also be "snapped" to a position (i.e., location and orientation) that can be defined by a vertex, surface, a coordinate system of another Device model in the layout, or a point on the floor of the layout. When a vertex, a coordinate system, or the floor is used in the layout to snap a Device model, the base origin of the Device model will be relocated on the selected vertex, the origin of the coordinate system, or the point on the floor, respectively, without changing the orientation of the base coordinate system of the snapped Device model. However, if a surface is used to snap a Device model, the Device model is to be snapped to a selected position on the surface with the *z*-axis of its base coordinate system aligned with surface normal.

The IGRIP Workcell Page also provides the functions for attaching a Device model to a parent Device model. To do this, an available coordinate system of a Part model on the parent Device model must be selected and used to superpose the base coordinate system of the attached Device model. By default, the base coordinate system of the selected Part model on the parent Device model is to be used for attaching the selected Device model. Once attached, the coordinates of the attached Device model will be with respect to the selected Part model on the parent Device model. Whenever the parent Part model moves, the attached Device model will move with it. For example, if a gripper Device model is attached to a robot Device model via the mounting Part model of the robot Device model, the gripper Device model becomes a Part model of the robot Device model. Thus, the gripper Device model moves with respect to the base coordinate system of the robot Device model. In this case, the TCP of the robot Device model can also be redefined on the attached gripper Device model.

6 DEFINING DEVICES' MOTION DESTINATIONS IN IGRIP LAYOUT

In the IGRIP layout, the motion destination position of a Device model is represented by a tag point that is a Cartesian coordinate frame with *n*, *o*, and *a* (or *x*, *y*, and *z*) axes. To determine the position of a tag point in the layout, a Part model of a Device model in the layout must be used to attach the tag point. With the attachment, the position of a tag point can be determined with respect to the base coordinate system of the Part model of the Device model to which the tag point is attached. In the database, all created tag points in the layout are stored in different named (tag) paths. A named path is an ordered collection of tag points that are attached to a common Device model. IGRIP allows a Device model to attach different named (tag) paths.

In the IGRIP layout, tag points are used as the TCP motion destinations of a Device model that has inverse kinematics. As a Device model with inverse kinematics is instructed to move to a tag point through various IGRIP methods (i.e., jogging or a GSL motion statement), the *x*, *y*, and *z* axes of the TCP frame of the model is attempted to superpose onto the *n*, *o*, and *a* axes of the tag point. If the relative TCP frame position of the model and the tag point position make this superposing impossible, the tag point is defined as unreachable one with respect to the Device model.

The procedures for creating and manipulating tag points in the IGRIP layout are:

Step 1 Creating a path for tag points in the database. To do this, a geometric model such as a Part model of a Device model must be first selected. Then, a unique name must be given to the created path.

Step 2 Creating tag points for a named path. There are three basic methods for creating tag points. The "single" method creates a tag point one at a time for a named path. When such a tag point is created in the workcell, it always first appears at the base origin of the device model that the named group is attached to. The "automatic" method is similar to the "single" method in the way that it creates tag points one at a time. The difference is that the "automatic" method automatically creates and names a new tag point for the named path. The new created tag point appears at the same position (location and orientation) of the current tag point in the workcell. The "multiple" method allows creating a number of tag points for a named path at one time. These tag points won't appear in the workcell until manipulation procedures such as "snap" are performed to them.

Step 3 Manipulating tag points in the layout. Once a tag point is created in a named path, it can be manipulated in many ways. Besides typical manipulation functions such as selection, translation, and/or rotation, the IGRIP Layout Context provides a variety of "snap" functions. These "snap" functions allow a user to place a created tag point by snapping it to a position defined by a vertex, edge, frame, curve, and surface of a Part or Device model appeared in the layout world. Constraints and options for a specific snap function can be set up before a snap function is performed. For example, if the "center" option is chosen, a tag point will be snapped on the "center" of the geometric entities such as line, edge, polygon, etc. If a tag point is required to snap on "surface," the parameter "approach axis" must be set up to determine which axis of the tag point will be aligned with the surface normal vector.

7 DEVICE BEHAVIOR AND PROGRAMMING IN IGRIP LAYOUT

Three types of movements represent the behavior of a Device model in the layout: action, motion, and manipulation. In IGRIP, the GSL statements in the GSL program of a Device model control the Device model for performing a series of movements in the workcell. In order to cause a harmonious behavior of a Device model in the layout, the Device's GSL program also realizes the interlock conditions through the signal passing on the Device's input/output (I/O) ports.

7.1 Device Action and Programming

A Device's action represents a simple movement generated by a joint of the Device model. Opening and closing the fingers of a gripper Device model are the typical example of this type action. Because an action is a result of a joint's movement with respect to the original or current position of a Device model, thus the Device model does not need to have inverse kinematics for generating a joint's action.

In GSL, the MOVE JOINT statements are used to realize Device's joint actions. Besides controlling the speed and travel distance of a joint, the statements may also be specified to control the joints' movements simultaneously, sequentially, and immediately.

7.2 Device Motion and Programming

A Device's motion represents a movement of the base frame or TCP frame of a Device model to a tag point in the

layout world. To generate this type of motion, the Device model must have inverse kinematics. In this category, a manipulator model represents the general case, where the TCP frame of the manipulator model is required to move to a tag point defined in the workcell.

A Device's motion is defined from the time it starts to the time it stops, which may consist of multiple motion segments. A motion segment is a single movement of the TCP and characterized by (1) initial and final tag point, (2) acceleration/deceleration, (3) commanded speed, and (4) trajectory. The control and simulation of a Device' motion in IGRIP involve in both the program developed for the Device model and the motion controller implemented in IGRIP software.

GSL Motion statements are used to simulate the movement of the Device being programmed. Among them are:

- Move Away Statement
- Move Near Statement
- Move To Statement

7.3 Device's Manipulation and Programming

A Device's manipulation represents a movement caused by the direct interactions of Device models in layout world. Typical examples include holding, pushing, pulling of a Device model by other Device models in the layout. In IGRIP robotic workcell simulation, it is sure that a Device's manipulation and programming must be applied to a robot's gripper Device model in order for holding and releasing a workpiece Device model.

In GSL, the GRAB and RELEASE Statements are used to deal with a Device's manipulation and programming. Unlike an "Attach" function, a "Grab" function or GRAB statements in GSL do not move the grabbed Device model at the time it is grabbed in the layout. However, like "Attach" function, once grabbed, the movement of the grabbed Device model follows the movement of the grabbing Part model of the grabbing Device model in the layout.

7.4 Device's I/O Communication and Programming

The I/O Page in the Layout Context contains functions that allow a user to establish I/O port connections between Device models in the layout. These I/O port connections are used to direct the flow of control signals during a workcell simulation run. The I/O port connections may be analog or digital. The number and type of I/O signals associated with a Device model are defined as the Device attributes. These attributes are set when a Device model is created, and can be changed during model development.

To set up the I/O port connections between two Device models in the layout, "Dual Connection" function can be used in the I/O Page. The user will be asked to select the first Device model for dual I/O port connections. Once selected, a list of I/O port names for that Device model will appear. The user should then selected one of the names marked "free." Once a "free" port has been selected for the first Device model, the user should repeat the same steps for defining I/O signals for the second Device model. The GSL I/O statements are to be used in the GSL program to control the passing of signals via I/O connections of a Device model.

8 EXECUTING IGRIP WORKCELL SIMULATION AND ANALYSIS

Given a workcell layout model with one or more Device models being programmed, the behavior of each programmed Device model in the layout can be simulated over time. This is achieved through simulation functions provided by the IGRIP Motion Context. There are steps that must be performed in order to conduct an IGRIP simulation run to a workcell model.

Step 1 Load GSL programs for the Device model that is to participate in simulation. In this step, the developed program for a device model can be selected from the IGRIP program directory and downloaded into the Device model in the layout.

Step 2 Active each Device model that has a loaded GSL program.

Step 3 Set simulation step size. A simulation step can be best described as a time slice of the simulation in workcell time. The unit of the simulation step is seconds. Updating the graphic display on the screen to show the Device models in their new position(s) can be performed at the completion of one step or a number of steps. In this sense, the smaller the simulation step is, the more detailed the graphic display is. For example, if there is great concern about collisions at some tag point during a simulation run, the step size should be set to a relative small value to assure that the collision-checking algorithm does not miss a collision display between steps.

Step 4 Conduct a simulation run. In this step, the initial configuration of Device models in the layout can be set before simulation begins. In addition, the "Run" types of simulation determine how the simulation stops. For example, a simulation run may continue until it finishes, encounters an error, or reaches the modeler-specified stop time.

During a simulation run, the GSL program that runs a Device model in the layout can be debugged, and performance data can be recorded and displayed. The common performance data includes Device's cycle times, joint values, joint speeds and accelerations, TCP values and speeds, I/O values, etc. More advanced procedures can be conducted as well for analyzing the performance of the workcell layout model. For example, an overlooked collision is one of the most important concerns in robotic workcell design. During a simulation run, all potential collisions in the layout can be automatically detected by the simulation system. This is achieved by (1) defining the pairs of Device models or pairs of Part models that may collide to each other in the layout, and (2) storing them into the collision queue of the simulation system. When the collision-checking function is "ON" during a simulation run, the simulation system will check exact collisions for all the Device pairs and Part pairs in the established collision queue. If a collision is detected, the simulation system may highlighted the collision and halt the simulation according to the user request.

9 CONCLUSION

Robotic workcell simulation is a modeling-based problem solving approach that fundamentally changes the way for designing a robotic workcell. This research has investigated a methodology for conducting robotic workcell simulation via Deneb's IGRIP robotic simulation technology. It shows that developing faithful models for a robotic workcell simulation study is a complex process that requires the multifaceted knowledge in diverse disciplines. With the developed methodology presented in this paper, designers in the field of engineering design and automation are able to easily understand the state-of-the-art technologies of robotic workcell simulation, and integrate these technologies with their knowledge and experience about CAD, robotics, and robotic workcell design.

REFERENCES

Chen, D. and F. Cheng. 2000. Integration of product and process development using rapid prototyping and workcell simulation technologies. *Journal of Industrial Technology*, Vol. 16, No. 1. pp. 2-5.

Craig, J.J. 1997. Simulation-based robot cell design in AdeptRapid. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, ICRA, Albuquerue, April, Vol. 4. pp. 3214 -3219.

Knasinski, A.B. 1997. Linking simulation to the real world through robot metrology. In *Proceedings of Robotic Simulation & Off-Line Programming Workshop*, Seattle, Washington.

Paul, R.P. 1981. *Robot Manipulators*, 8th Ed., The MIT Press.

Rooks, B.W. 1997. Offline programming: a success for the automotive industry. *Industrial Robot* 24(1), pp. 30-34.

Rubinovitz, J. 1999. CAD and graphic simulators /emulators of robotic systems. In *Handbook of Industrial Robotics*, S.Y. Nof (eds.), John Wiley & Sons, Inc, pp.755-773.

Rudnick, F.C. and S.G. Moore. 1997. Robotic paint simulation and off-line programming in a rapid prototyping environment. In *Proceedings of Robotic Simulation & Off-Line Programming Workshop*, Seattle, Washington.

AUTHOR BIOGRAPHY

FRANK S. CHENG is an assistant professor in the Department of Industrial and Engineering Technology (IET) at Central Michigan University (CMU). He holds a M.S. degree in Mechanical Engineering, and Ph.D. in Industrial Engineering from the University of Cincinnati. His research interests include robotic workcell design and simulation, Petri nets, flexible automation, and control system design. He is a member of SME, NAIT, and ASEE. He is currently engaged in the industry and NSF sponsored projects in developing methodologies for teaching and applying robotic and manufacturing simulation technologies. His email is <fcheng@iet.cmich.edu>.