SOFTWARE-BASED DESIGN AND SIMULATION OF ROBOTIC ASSEMBLY SYSTEMS

Pierre Dumuid and Lachlan Smith

Department of Mechanical Engineering, The University of Adelaide
Adelaide SA 5005, Australia

ABSTRACT

The aim of this project was to analyse software-based techniques for designing and simulating robotic assembly systems, and to draw conclusions on the suitability and advantages of simulation in such environments. Despite a large amount of literature being available regarding simulation in manufacturing, there was little specifically related to that of robotic workcells. It was therefore considered a major aim of the project to draw conclusions specifically related to the simulation of robots, not just manufacturing systems in general. These aims were implemented firstly by conducting extensive research into existing simulation methods. A leading software package in this area, IGRIP, was then analysed to determine how such systems operated.

In order to understand fully the concept of simulation, a case study based on consulting work recently undertaken by the University was implemented. A fully operational workcell was created using the IGRIP software. To achieve this it was necessary to be extremely familiar with all aspects of the software, from detailed CAD work to complex programming. Implementing this workcell placed us in an excellent position from which to assess the capabilities and advantages of such simulations, as well as their limitations.

KEYWORDS: Robots; Simulation; Design; IGRIP.

1. INTRODUCTION TO SOFTWARE-BASED DESIGN AND SIMULATION

Historically, manufacturing plants have been satisfactorily designed without the aid of computer simulation. Hence, if simulation is to be recognised as a viable tool it must either provide better manufacturing designs or offer cost benefits in engineering designs or installation efforts. The experience of many firms has been that using simulation does indeed realise these benefits [1].

The acknowledged advantages of simulation can be summarised as follows: versatility – computer modelling may be used to represent a wide variety of real-world systems; flexibility – computer models may easily be altered to represent different situations of updated information; cost effectiveness – experiments using computer simulation enable the performance of a system to be reliably investigated without building the physical system; non-disruptive – simulation experiments permit a system to be designed, redesigned and analysed without disrupting any existing system; exhaustive – simulation experiments may be performed under every conceivable set of system conditions, parameters or operating characteristics [1].

Industrial simulation has been characterised by two trends in recent years. The first is the increasing use of simulation with respect to complex manufacturing systems. The second is the increasing use of computer graphics for animated displays of the movement of entities through the simulated system [2]. In addition, with the increasing level of automation in modern industry, robots have become an integral part of most manufacturing systems. Software capable of modelling robots is therefore a requirement for simulating such systems.

In general, robots are used for one of two purposes: as a productive workstation, such as welding or assembly, or as a means of handling components, such as unloading and loading machines [3]. In assembly operations, ancillary equipment such as bowl feeders, waffle tray feeders etc. will probably be present. In a simulation model it will not normally be necessary to model these ancillary features in detail, only to model the components on which there is a specific focus, such as the robots themselves.

2. IGRIP ROBOT SIMULATION SOFTWARE

IGRIP stands for Interactive Graphics Robot Instruction Program. It was originally developed by Deneb Robotics (now Delmia), a subsidiary of the renowned Dassault Systemes, and is a robotic simulation package that incorporates real-world robotic and peripheral equipment, motion attributes, kinematics, dynamics and I/O logic. It is described by its manufacturer as a physics-based, scalable robotic simulation solution for modelling and off-line programming of complex, multi-device robotic workcells [4].

Put simply, the basic package, as used for the purposes of this project, allows designers to build devices from scratch, assign kinematics to these devices, place them in a workcell, and program them to perform complex operations. One of the major features of the package is the realistic, high quality, three-dimensional animations of extremely complex systems it is capable of producing. The software itself contains a library of robots produced by the major manufacturers, which can if necessary be preloaded into the workcell.

A recent example of this software be used in industry was its use by Boeing in the design of one of the world’s largest robots, used for riveting during the construction of the C-17 transport aircraft.

The designer interface menu system is divided into a series of 'contexts'. The operation of these contexts is described below. An alternative to the menu system is the Command Line Interpreter (CLI), where operations are invoked by text commands instead of the mouse.

2.1 The 'CAD' Context:

This is where the initial device modelling takes place. In general, 3D modelling can be divided into two types: boundary representation and constructive solid geometry [5]. IGRIP employs the latter. Models are constructed from solid primitives such as blocks, cones, spheres, cylinders and hexahedra. Boolean operations, such as addition, subtraction, union and intersection are then applied to these objects. It was found that the most complex operation in the CAD context is determining the spatial relationships between objects. Whenever an object is created, aside from specifying its dimensions, its global position as well as the location of its base coordinate system must be specified.

2.2 The 'Device' and 'Layout' Contexts:

Before a part can be assigned kinematics, it must be 'recreated' as a device. This action is performed in the device and layout contexts. On importing a part from the CAD context, IGRIP instructs you to define parameters such as its name, degrees of freedom, programmability, global reference frame and the number of digital input and output signals. New devices can be attached to existing devices as subdevices, and IGRIP will automatically recalculate the inverse kinematics of the extended device. The type of motion (rotation, translation etc.), home positions and limits of travel can also be defined.

2.3 The ‘Motion’ and ‘Program’ Contexts:

A subset of the layout context is used for defining paths of motion for each device. On each path are placed a series of 'tag' points to which devices (such as robot end-effectors) can be moved. The orientation of the end-effector at a tag point is set by the orientation of the frame attached to that tag point, and the order in which tag points are reached is defined by the device's program. The 'Motion' context itself is used for starting and controlling simulations and animation. The 'Program' context is, as the name implies used for developing device code. It was found during the course of the project that it was faster to code in a Unix text editor, such as 'emacs', then import it into the software.

2.4 Programming of Devices

```
PROGRAM pierreGrip7Aug
VAR ...

...
ROUTINE nextbox(...)
...
END nextbox
...
BEGIN MAIN
  ...
  $SPEED = 10000
  $MOTYPE = SLEW
  DOUT[ 1 ] = 0
  MOVE HOME
  FOR boxno = 1 TO 4 BY +1 DO
    gotbox = nextbox('pkentrybay', ...)
    MOVE TO pkfill_frentry1
    MOVE TO pkfill_drag
    MOVE TO pkfill_pick
    drop_box()
    MOVE TO pkfill_wait
    DOUT[ 1 ] = 1
    WAIT UNTIL DIN[ 1 ] == ON
    DOUT[ 1 ] = 0
    WAIT UNTIL DIN[ 1 ] == OFF
    MOVE TO pkfill_pick
    grab_box()
    MOVE TO pkfill_drag
    MOVE TO pkfill_toexit1
    gotbox = nextbox('pkexitbay',...)
  ENDFOR
  MOVE HOME
```

Figure 1: Example Program

Figure 1 shows a typical program layout and communication structure that is used in encoding devices. It is part of a larger program used for controlling a palletising robot (see case study, below). Every device that is assigned kinematics requires code to control its movements and allow it to interact with other devices. The language used to program the devices is very similar to that used for physical robots, and facilitates a transfer from the simulated world to the physical world, with minimal changes. As well as having loaded programs, devices have digital I/O links to define how they interact.

Code is written in Graphic Simulation Language (GSL), a language which allows the use of compact subroutines to perform complicated functions. Motion parameters are set using various commands beginning with the $ symbol, the *nextbox* routine calculates the position of the next box to be picked up and the control routine is used to communicate with the grippers. The grippers themselves also have a routine to attach and detach the devices. The opening and closing of the grippers is controlled by the *grab_box()* routine, defined elsewhere in the program.

These routines allow the passing of variables between them and the main program, which provides more error handling and control. To cause the robot to move between different positions, the command, *MOVE TO* is used followed by either a tag name, (e.g. pkfill_drag) or a string that evaluates to a tag name. As well as the general *MOVE TO* command, there are various other commands that allow movement of the robot by joint, relative movement (relative to a tag orientation) and *MOVE NEAR* commands. Additional commands that are impossible in GSL can be called in CLI.

Other Features:

The ‘ARC’ context is used to invoke add-on applications, used for such things as painting, welding and finishing. The ‘Draw’ context is used to design in a two-dimensional world instead of a three-dimensional world. The ‘User’ context is used to customise the menu system. The ‘Analysis’ context is used to determine the properties (such as dimensions) of objects in the workspace. The 'SYS' context provides the ability to define system properties, such as views, colours etc.

IGRIP is available for both Unix and Windows environments. The Unix version was used for this project.

3. CASE STUDY – IPLEX PIPELINES

This case study is designed to demonstrate the complete design process of an automated production system using the IGRIP software. It is based on a consultancy project recently undertaken by Adelaide University for IPLEX Pipelines, and the design has been completed to such an extent that it could be physically implemented with only a minimal amount of additional work.

3.1 System Specification

The existing workcell at the factory comprises of a series of plastic pipe fittings being manually placed on a conveyor, passing a printer which places a bar code on each, then falling into a box at the end of the conveyor. The workcell is to be redesigned so that the process is fully automated. In addition, the system must be capable of rotating components during the printing process so that barcodes can be printed on components of smaller diameter.

3.2 Theory

The criteria for selecting the best way to implement the required system were divided into two categories. The first set is based solely on such factors such as efficiency, reliability and cost. The second took these into account, but also examined the capabilities of the software, and how its primary features could best be demonstrated and assessed.

At the very least, it was concluded that the system would have to include a feeding mechanism, a parts identification system, a component handling system, a printer and component packaging (most likely a box of some kind). In addition, it was decided to include a simple palletising system at the end of the production line.

Several initial concepts were developed, each containing varying features, from conveyors to rotating printer heads, vision systems, light sensor arrays and, of course, robots.

3.3 The Chosen System

Based on the afore mentioned criteria, the following system was developed for implementation in IGRIP. Components enter on a conveyer in waffle-trays, so that they are at known positions and orientations and can easily be located by a component-handling robot. The robot chosen for this task was a Motoman KS3 due it is suitable size and workspace. A tool capable of internally grabbing each component and axially rotating it, transfers components one at a time to a printer, situated adjacent to the conveyer, then drops them into a waiting box. Waffle trays are detected optically by a sensor at the end of the conveyer, which also determines the type of component being handled by reading a barcode on the side of the waffle-tray. A simple actuator pushes empty trays to a second conveyor, which removes them from the workcell.

A Fanuc M410i robot was deemed most suitable for palletising. It would be responsible for transferring empty boxes to the loading station, and completed boxes to a dispatch area. An optional addition would be a packaging robot (most probably a SCARA robot), for placing lids on boxes. It was apparent that this was not the most efficient or likely packaging method, however, to simulate a more complex packaging routine in IGRIP, though possible, would have required extremely complex coding. It was decided that the time required to learn and implement such code would be better spent focussing on the robotic elements of the system.

While this system contains many robotic elements, and would thus be comparatively expensive to implement, it was deemed extremely efficient and reliable. Most importantly, it would demonstrate the capabilities of the IGRIP software package very well.

A layout schematic of the proposed workcell is shown in Figure 2.

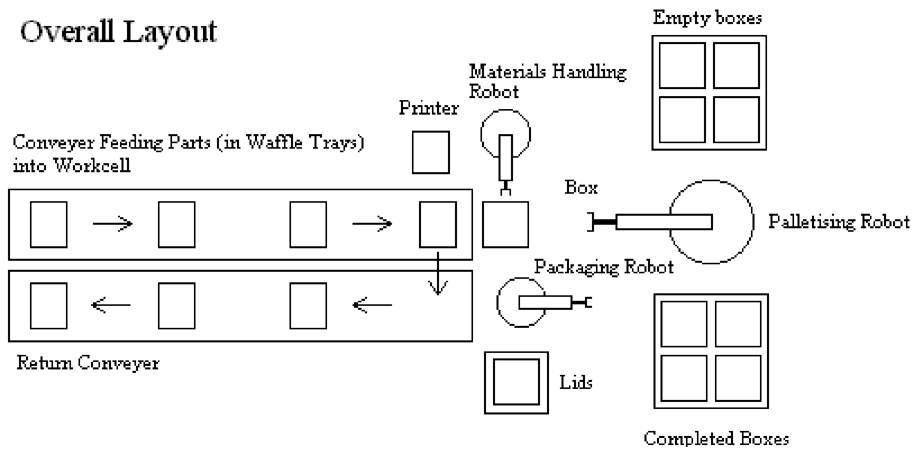

Figure 2: Layout Schematic of the Proposed Workcell

3.4 Implementation

The following is a description of how the system defined above was designed and simulated using IGRIP. A screen capture of the completed workcell in operation is shown in Figure 3.

All CAD work was carried out first. This required the completed design of the following components: conveyors; printer; waffle-trays (with barcode); parts to be processed; barcode reader and sensor; component manipulation end effector; palletising end effector; boxes; loading station base; pallet bases; waffle-tray actuator; robot bases.

The next step was to create devices from these parts. The devices that required kinematics were the waffle-tray actuator, both end effectors and the robots themselves. It was not necessary to assign kinematics to the waffle-trays or conveyors, as it was found that these could be more efficiently moved using the ‘shift’ command.

The robots were then imported into the workcell, and the end effectors attached to their mounting plates, automatically redefining their kinematics. All static devices were then placed in the workcell, and all devices referenced to the global coordinate system.

An efficient GSL routine for operating the palletiser was then developed. This code was written in such a way so that it could also be used with the component handing robot, with only minimal alterations. All relevant tag points were then placed in the workspace, being careful that all were at the correct orientation, and all were inside the reachable workspace of the robots. The powerful ‘RayCast’ command was used to determine distances between robot end effectors and the components they are required to grab.

Communication links were then developed between devices. For example, the component handling robot was told to wait until a box was at the loading station before commencing its operation, and the palletiser was told that when a box was full, it had to be removed and replaced by an empty one.

Finally, the optimal operating speeds of the devices were determined. It was possible to generate graphs of robot joint velocities to ensure all devices were operating efficiently and safely.

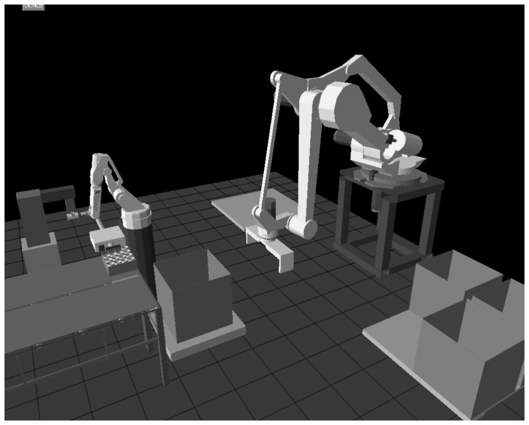
Figure 3: Screen Capture Showing the Workcell Created in IGRIP

4. CONCLUSIONS

Several important conclusions can be drawn from the outcome of this project. Firstly, it is apparent that simulation packages such as IGRIP provide designers of robotic systems with an invaluable visual tool for determining the exact performance of systems without the aid of any physical models. In addition, the powerful nature of Graphic Simulation Language and its ability to mimic actual physical systems allows designers to cater for all possible design scenarios, and adapt and implement an infinite number of design changes with only minimal additional work. In fact, a recently released update allows designers to physically interact with their creations via a virtual reality interface. The major drawback for a company considering the purchase of this software, aside from its high cost, would be the extensive training in its use required for designers.

Secondly, it is apparent that such software could potentially be an excellent teaching tool in Universities, as it would allow students to move beyond basic theory, and actually implement and visualise their designs.

Finally, the authors sincerely hope that the work carried out during the course of this project will be a valuable source of information in coming years both in industry and the university.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Cheryl Li, Dr Anthony Zander and Mr Antoni Blazewicz for their assistance at various stages of this project.

REFERENCES

[1] R. Miller, *Manufacturing Simulation*, Lilburn: Fairmont Press (1990)

[2] J. Grant and S. Weiner, *'Factors to Consider in Choosing a Graphically Animated Simulation System', Industrial Engineering*, August 1996 p.1

[3] A. Carrie, *Simulation of Manufacturing Systems*, Essex: Wiley & Sons (1988)

[4] *IGRIP User Manual and Tutorials*, Deneb Robotics, Inc., (1996)

[5] S. Meguid, *Integrated Computer-Aided Design of Mechanical Systems*, New York: Elsevier Science Publishing Co. (1987)