Journal of Materials Processing Technology 139 (2003) 8–14

Journal of Materials Processing Technology

www.elsevier.com/locate/jmatprotec

Post-processing methodologies for off-line robot programming within computer integrated manufacture

S.F. Chan*, Reggie Kwan

School of Science and Technology, The Open University of Hong Kong, Room A0911, 9/F, 30 Good Shepherd Street, Ho Man Tin, Kowloon, Hong Kong, PR China

Abstract

A robot simulation system has been used not only as a design and simulation tool, but it also serves as an off-line programming system for industrial robots and other machines with a computer integrated manufacture (CIM) system. A robot simulation system is one of the essential elements for virtual manufacturing and agile manufacturing. A post-processor is an important element of an off-line robot programming system, as it allows the commercially available robot simulation systems to be used as off-line robot programming tools. The post-processor translates output statements from a robot simulator to a target robot language. Typically, the sequence of motion and data required to drive the robot can be transmitted through the use of a serial or parallel data link, between an off-line computer and a robot controller. The design of a general post-processor capable of translating the output of multiple robot simulators to multiple robot programming languages is an extremely complex problem.

There are two basic ways of describing robot movements in robot control systems; one describes the manipulator movements in terms of the manipulator end-effector location (in compound transformation or absolute); whilst the other describes the movements in terms of manipulator joint angles. The former approach is suitable for future and present generation robots with a language processor installed in their controllers, whilst the latter is only specifically linked to earlier generation robot languages. Discrepancies between a specific robot simulation system and a real robot system which should be resolved as part of the post-processing, are analysed. The feasibility of generalising the post-processor was studied. In this paper different approaches of post-processing for off-line programs and methods of downloading off-line programs to robot controller are discussed also.

Keywords: CAD; Robot simulation; Off-line robot programming; Post-processing methodologies; Generalised approach; Program download

1. Introduction

The post-processor is an important element of an off-line robot programming system, as it allows the commercially available robot simulators (design tools) to be used as off-line robot programming tools. The post-processor translates output statements from a robot simulator to a target robot language. Typically, the sequence of motion and data required to drive the robot can be transmitted through the use of a serial or parallel data link, between an off-line computer (where the proposed system is usually designed and simulated) and a robot controller. When using contemporary systems the post-processor translates from the model output into a vendor specific robot language, and extracts location information from the simulation model. The design of a general post-processor which is capable of translating the output of multiple robot simulators to multiple robot programming languages is an extremely complex problem and requires further consideration. Fig. 1 describes a general off-line robot programming system.

There are two basic ways of describing robot movements in robot control systems; one describes the manipulator movements in terms of the manipulator end-effector location (in compound transformation or absolute); whilst the other describes the movements in terms of manipulator joint angles. The former approach is suitable for future and present generation robots with a language processor installed in their controllers, whilst the latter is only specifically linked to earlier generation robot languages. The advantages and disadvantages of using on-line versus off-line robot programming have been discussed widely [1–7].

Some recent development work includes the includes the V^+ operating system and the application information management (AIM) software from Adept Technology Inc. [8,9],

* Corresponding author.
E-mail address: sfchan@ouhk.edu.hk (S.F. Chan).

0924-0136/03/$ – see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0924-0136(03)00174-2

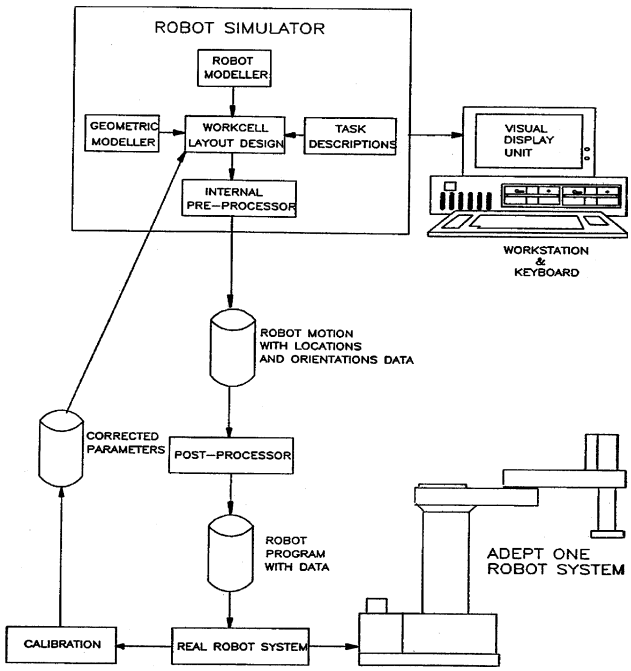

Fig. 1. General off-line robot programming system.

and is an example of vertical integration from the robot manufacturers. Whilst V^+ continues as the foundation of the Adept System evolved from its previous programming language VAL II, AIM provides a graphical user interface for ease of use. Although the integrated system of V^+ and AIM provides a powerful programming environment for off-line robot programming and real-time control, the functionality is system-dependent. The digital factory focuses upon detailed simulation of manufacturing processes as well as robot programming based on the foundation of numerical control programming [10]. There has been a number of industrial organisations involving the use of a robot simulation system for different robots and applications. Some typical efforts can be cited as good examples of the wide applications of off-line robot programming, including the following.

1.1. Aerospace welding

The Rockwell Science Centre in California of USA, used the McDonnell Douglas off-line robot programming system to design and program robot welding cells [11] for welding space shuttle main engines [12]. NASA engineers programmed the operation even before the equipment arrived [13], based on the welding parameter database specified by welding engineers and on CAD databases that describe the parts to be welded. Each engine has 14,000 in. of weld and a variety of weld geometry. Obviously, on-line teaching would require extensive debugging and lengthy downtime. The off-line robot programming system holds a weld parameter database, which is processed to facilitate programming of robotic tungsten arc welding.

1.2. Automobile industry—spray painting

Robot programming for spray painting applications is almost exclusively done by manual guiding methods. This has serious drawbacks since the worker cannot produce the best quality of coating (being hindered by the weight and stiffness of the robot arm and skill of the programmer) especially in the case of large surfaces [14] such as bus bodies [15,16]. Advances in CADCAM methods and a growing interest in off-line robot programming led to this technique being applied for painting robots [17], which led to a better quality of coating and a decrease of paint loss. The approach involves modelling the spatial distribution of the paint particles. Theoretically, when the workpiece has large surfaces which are planar or close to planar and the surface normal changes regularly, automatic path generation is possible. Based on this a workpiece of arbitrary shape can also be painted by segmenting the workpiece into planar patches; that is, curved surfaces are approximated by polyhedra. From their experimentation, the efficiency of the painting process increases with the size of the surface to be painted. Efficiencies of 70–80% have been achieved through off-line programming which are better than with current practice where a 55–60% efficiency is typical (efficiency is defined as being the ratio of paint usefully applied to the total consumed).

1.3. Automobile industry—spot welding

The General Motors Corporation plant in Doraville, USA, applied off-line robot programming for automotive spot welding [18]. This involves the use of the IGRIP simulation system to model the body framing station (a station that holds the car body in place ready for welding to proceed). The framing station includes six GMP S-480R robots, with two arranged on either side of the car body, and one at the front and rear. During the framing and spot welding process, the car body is driven into the station by an automatic guided vehicle (AGV), and is locked in place by fixtures. The robot then performs the required welding operations, and after the fixture release the AGV moves out of the station. It was quoted that approximately 60 h were required to generate the robot programs through simulation and off-1ine robot programming whereas 300 h were required when using on-line methods.

1.4. Glass cutting

Nottingham University has investigated the automation of the cutting of patterns on glassware [19,20], by generating off-line robot programs from a CAD system. Once the patterns are designed, the surface profile of each glass is measured using sensors so as to determine a bi-cubic surface patch model of the glass surface. The pattern definition from the CAD system is then mapped onto this surface patch in order to calibrate the robot program. This robot program is

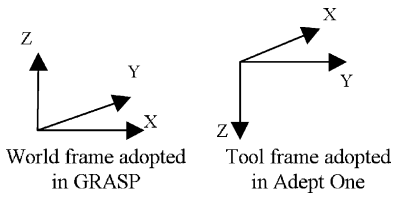

Fig. 2. Difference in frame concepts between GRASP and Adept One robot systems.

tailored to an individual glass and hence the whole process must be repeated for each blank glass.

1.5. Electronic component insertion on PCB

Research work has been carried out to integrate product data captured in CAD database with a robot simulation system whereby producing a simulation model and its associate robot program for assembly task off-line. The post-processing methodologies were studied and the off-line robot programs generated can be re-converted back into acceptable language readable by the robot simulation system together with calibrated date about the robotic cell for assembly [21].

In this paper, the authors will discuss different methodologies for post-processing and its generalisation. Examples will be cited with reference to the chosen robot simulator (GRASP), target robots and methods for dealing with peripheral devices such as sensors, vision system and tooling, and the functionality available from the corresponding robot language (VAL II, a predecessor of V^+).

2. Discrepancies between robot simulators and real robot systems

There are a number of facilities available in robot simulators which are not available in real robot languages and vice versa (e.g. the real operation of sensors and vision systems cannot be easily simulated and instead the robot simulators use control conditions to emulate sensors and vision systems). Whereas the coordinate frame concept is used in most simulators, the majority of robot simulators use an Eulerian angle set [22] for any kinematic (direct forward and inverse) calculation. However, in the case of the Adept One robot, although the Eulerian angle set is used, the coordinate frame uses a right-handed axis set but its tool reference frame is rotated through 180° about the *Y*-axis and 90° about the rotated *Z*-axis, as shown in Fig. 2.

In order to overcome this discrepancy, the tool attachment point of the robot has to be rotated about the *Y*-axis by 180° in the robot simulation model. This leads to the programming of the robot TRACK (a specific name used in the GRASP system which comprises event-dependant motion sequences) to include the rotation of 180° about the *Y*-axis (except all locate statements). In simulation, the robot end-effector's coordinate frame is the coordinate frame of any tool or object currently attached to the robot flange. However, in real robot systems, the robot end-effector is always referred to as the robot flange. The difference between these two frames should be corrected by using the TOOL statement in VAL II. Nevertheless, the translation between outputs from a robot simulator and a robot language can be made into a one-to-one conversion.

3. Theories used in post-processing for different generations of robots

Early generation robots such as the ASEA IRB6, do not include a robot programming language, but use very simple command statements and condition looping. Such a robot can only take joint angles corresponding to individual robot joints and so the post-processor must be capable of converting simulation output from Cartesian coordinates into individual robot joint angles. The robot controller is not sophisticated enough to deal with high level languages and the data communication capabilities for 'supervisory' systems are too simple for data transmission, although a microcomputer can be used as an external supervisor which can cope with high level to lower level robot language conversion.

Modern robot controllers are likely to incorporate a language processor such that the robot movement can be driven by language command statements and location data. This makes the development of a post-processor easier. This type of post-processing involves a conversion of the simulated sequences of motion into corresponding robot language statements. This approach was used for the development of the post-processor which is discussed later.

4. General methodologies of post-processing for off-line programs

Post-processing facilities can be system-dependent, application-dependent or generic. System-dependent post-processors are the most commonly used, and function by interpreting and translating program statements of a robot simulator into a specific robot language. In other words, system-dependent post-processors are specific to one robot simulator and one robot language. System-dependent post-processing facilities are the building blocks for application-dependent post-processors which are tailored specifically for a particular application with custom "macro" sequences. Generic post-processors are theoretically capable of translating the output of multiple robot simulators into robot languages for different robot controllers. Since robot languages differ from one another, the creation of a generic post-processor as described is not considered to be an achievable task. The most probable solutions for future implementation are thus expected to rely on standard data formats such as Industrial Robot Data

(IRDATA) [23] and Manufacturing Message Services (MMS) [24] or enabling technology such as Manufacturing Automation Protocol/Technical Office Protocol (MAP/TOP) [25,26].

Assuming access to the source code of a robot simulator, post-processing software modules can be written to modify certain data structures used by the robot simulator to provide new data structures as required by a particular robot. It will be seen from the following discussion that the person writing the post-processor may only have limited access to the various data structures used internally by the simulator. This ultimately can limit the available functionality of the post-processor. For example, in certain cases the vendor of the robot simulator provides limited access data structures via a so-called 'standard' interface. In such circumstances after a simulation exercise, internally created data structures can be post-processed into a robot-independent data format, which in the case of GRASP, is known as GRDATA. Here the post-processing function within the GRASP system outputs those entitles that have been referenced in the robot TRACK, the program logic and sequence and the gripper or tooling to be used. This is effectively fast post-processing since it does not need to post-process every entity in the simulation model. However, the GRDATA format cannot be used for any other purpose except for off-line robot program translation and to convert a VAL II program into a TRACK. This robot-independent data format (e.g. GRDATA) is then translated into a specific robot language (VAL II in this study) using a specific robot language translator. This approach of post-processing has the advantage of more efficient use of data, reduced processing time, and more importantly, and reduced chance of error (computation error) during the processing.

In general, users do not have access to source code of any simulation system supplied as suppliers are very security cautious. Without the supplier's source code, a system dependant post-processor can be created in a modular fashion (see Fig. 3). Access to the simulator's data structures through simulation model output. The first module extracts spatial relationships between entities in the workplace of a simulation model. This information is stored in a data file ready to be used as a reference data file. The second module of the post-processor translates motion sequences (TRACK statements) into a specific robot language (VAL II). The post-processing facility has been enhanced through two main stages: (a) Hierarchical Post-Processing, "Top-Down" approach and (b) Hierarchical Post-Processing, "Bottom-Up" approach. These two approaches were designed to coincide with the methodologies conventionally adopted in the construction of a CAD model and the way the spatial relationships between entities were described and stored. These approaches also represent a logical and efficient use of data captured in any CAD system. Discussions on the abovementioned two approaches will be made with reference to the previous study at Loughborough University, UK [21].

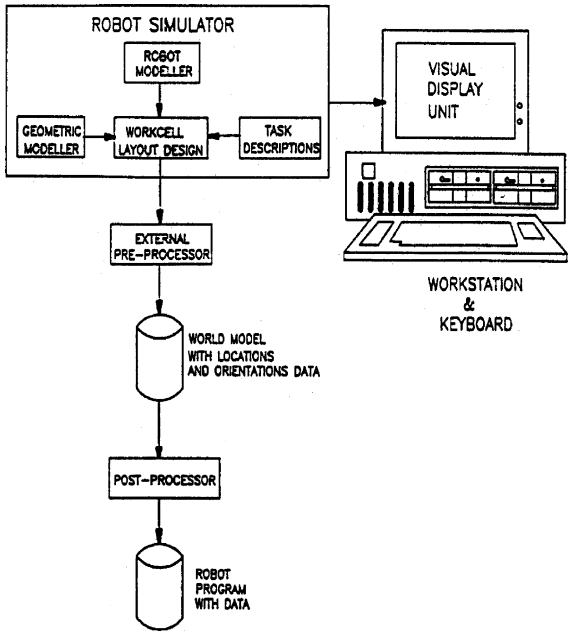

Fig. 3. Post-processing through external post-processors.

4.1. Hierarchical Post-Processing: "Top-Down" approach

CAD programmers are allowed to record move positions in either absolute form (i.e. relative to the global workplace coordinate frame) or relative form (i.e. relative to the coordinate frame of an entity within the model). This has the advantage of permitting the modification of the relative positions of entities in the model without the need to re-programme the task locations. This post-processor version comprises three modules coded in PASCAL.

The first of these modules, called PROCESSOR1, is used to extract the spatial relationships between entities and objects in the model. This information is stored in a data file, ready for use in the second module. PROCESSOR1 extracts information using a Top-Down approach (Fig. 4). The data file comprises random access data records (as shown in Table 1), and each record stores the name of the object and the name of its owner object, with their spatial relationships defined within a simulation model presented in a 4×4 matrix. These data can be used for calibration

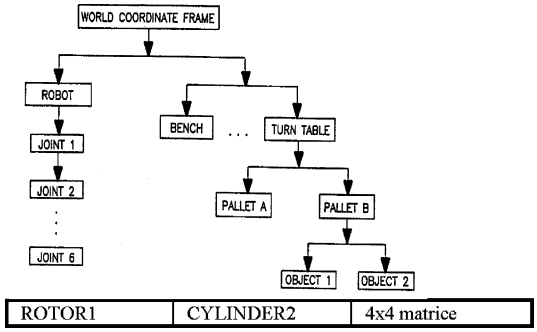

Fig. 4. Data structure of "Top-Down" approach.

Table 1
Example of random access date file

Owner object name	Object name	Spatial relationship
Workplace	Platform	4 × 4 matrix
Platform	Platform top	4 × 4 matrix
Platform	Stand	4 × 4 matrix
Platform	Rotor1	4 × 4 matrix
Rotor1	Cylinder1	4 × 4 matrix
Rotor1	Cylinder2	4 × 4 matrix

and updating of the CAD model. They can also be used as a neutral data format for data exchange between different robot simulators with similar data structures.

The second module, called PROCESSOR2, is used to extract from the robot TRACK the motion sequences, locations and orientations of the move positions relative to a particular entity in the simulation model. Since we are interested in the location and orientation of a position in space to which the robot should be driven, the locations and orientations of such positions are defined with reference to the robot base, e.g.

$$^{\text{robot}}_{\text{position}}T = [^{\text{workplace}}_{\text{robot}}T]^{-1} \times [^{\text{workplace}}_{\text{platform}}T] \times [^{\text{platform}}_{\text{position}}T]$$

This module searches through the data file produced in PROCESSOR1 for the name of the object and retrieves the 4×4 spatial matrix. This is multiplied by the transformation used in the TRACK movement. The data file is repeatedly searched and the matrices concatenated until the highest level is reached. The final location information is written in absolute coordinates relative to the robot base. The information representing the robot movement is then converted into a robot specific language called VAL II via VALII module which is similar to VALFORMATTING module as described earlier. The VAL II robot program and the location data are produced in one text. Since the robot program uses numbers to identify positions instead of the actual entity name used in the CAD model, the conversion of VAL II robot programs into GRASP TRACKs is not possible.

Example of output from PROCESSOR2 where absolute position for robot movement is used to drive the actual robot; calculated with respect to the robot base coordinate frame, e.g.

```
.PROGRAM EXAMPLE B
SET POINT1 = TRANS(200, 200, 800, 0, 180, 0)
SET POINT2 = TRANS(200, 200, 750, 0, 180, 0)
SET POINT3 = TRANS(500, 500, 800, 0, 180, 0)
SET POINT4 = TRANS(500, 500, 750, 0, 180, 0)
SET POINT3 = TRANS(500, 500, 800, 0, 180, 0)
SPEED 50 ALWAYS
MOVE POINT1
MOVE POINT2
CLOSEI
MOVE POINT2
MOVE POINT3
MOVE POINT4
OPENI
MOVE POINT4
MOVE POINT5
.END
```

4.2. Hierarchical Post-Processing—"Bottom-Up" approach

The post-processor produced comprises two modules. The first module, called PROCESSOR11, is quite similar to the first approach PROCESSOR1 module as described in Section 4.1, except that it is modified to increase the speed of processing such that the processing time is reduced to approximately 10% of that in stage 2 approach. This represents a significant saving, approximately up to 90% in the processing time. This module extracts spatial relationships concerning each object and its owner in the order of their appearance. The concept of "Bottom-UP" approach is illustrated in Fig. 5.

The second module of the post-processor is called PROCESSOR22, which translates the robot TRACK into a VAL II robot program with all the move positions stored in a separate text file. This allows compound transformations to be used in the robot program whereas in the previous stage only absolute positions were allowed. Such compound transformations is demonstrated as below:

MOVE REFERENCE:TRANSFORM

where REFERENCE is the location of the referenced object with respect to the robot base, i.e.

$$^{\text{robot}}_{\text{reference}}T = [^{\text{workplace}}_{\text{robot}}T]^{-1} \times [^{\text{workplace}}_{\text{platform}}T] \times [^{\text{platform}}_{\text{reference}}T]$$

TRANSFORM is the object's location with respect to the referenced object.

This method allows the off-line generated programs to be calibrated, the simulation model to be updated, data exchange between robot simulators with similar data structures may be implemented, and the actual robot programs can be

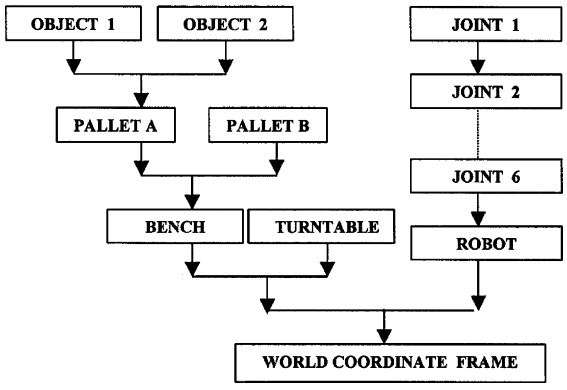

Fig. 5. Data structure of "Bottom-Up" approach.

reconverted into GRASP TRACKS. The format of actual robot programs can be reconverted into GRASP TRACKs by using the VALTOTRACK module created by the author. The VALTOTRACK module reads the robot program and its corresponding location data file which allows path control, movements, and gripper control statements to be converted into GRASP syntax. This VALTOTRACK module allows robot simulation systems to be used in a relatively comprehensive manner. For example, the verification of a robot program, for different workshop layout configurations and for different robots to perform the same task, or the same robot for different applications to be simulated and edited.

5. Methods of downloading off-line programs to the robot controller

When post-processing is complete, the program may be transmitted to the robot controller in one of following possible ways:

(a) The robot program can be downloaded via a robot specific medium (such as magnetic tapes, paper tapes, floppy diskettes, etc.) which can be transferred manually from the design office to the shopfloor. In the context of highly automated computer integrated manufacture (CIM) systems, this may require unwanted human intervention and may lead to increased machine set up times.
(b) If it is desired to load programs electronically through an available robot specific medium interface, then a communication protocol must be implemented in software in order to interface to the robot controller. The use of an RS-232 link to facilitate the communication between the CAD system and the robot controllers is quite commonly used in industry. However, this approach is too rigidly fixed to the specific machines and will result in a low level of portability (or configuration flexibility) as CIM systems become more commonplace and/or complex.
(c) A more flexible approach could be based upon the use of the MAP/TOP. This new enabling technology provides control data, monitoring status transmission, etc. and potentially there is a much reduced need to write specific protocol conversion for each robot. Furthermore, developments with respect to layer 7, the application layer, including the MMS which is being evolved in the wake of the MAP/TOP initiative should be monitored and could form the basis of a third generation command language.

MAP broadband network and advanced internet facilities enable users and programmers to transmit off-line robot programs and data to and from the off-line simulation workstation and the actual robot cell for calibration. This removes the barriers previously hindered the development in massive data transfer and obviously the reliability of the system. The internet facilities and web technology allow virtual teaming and virtual management to facilitate international collaboration and services.

6. Generalised approach

Post-processors can be system or application-dependent. Some of these post-processors have been designed for use with an application-dependent robot simulator. These post-processors normally have “macro sequences” built in for the particular application. However, most of the post-processors are system-dependent, which means they can only be used for one robot simulator and one robot language. If an internationally accepted standard data format existed, it would be logical to develop the post-processor in two modules. The first one being responsible for converting information from the output of a robot simulator to a neutral or standard data format, whilst the second module translates the standard data format into a robot language. This reduces the effort in developing a post-processor for each set of robot simulators and robot languages. This idea has been incorporated in the processors created and described in Section 4. However, standard data formats are not yet available. Thus a random access data file with consistent data structures has been used through providing a prototype neutral data format. Suppose, there are n robot simulators and m robot languages, then there is a need to write n pre-processors and m post-processors. However, with no internationally accepted standard data format (or neutral data format) there would be a need for $n \times m$ processors. This standard or neutral data format approach can be regarded as a generalised approach. There is a further possibility of generalising the post-processors so that common command statements are used in robot languages and post-processors.

The successful experience gained indicates that similar effort can be applied to different systems. Of course, the total effort will be significantly reduced if standards exist. However, at the moment, there is no standardisation that has been successfully achieved in the field of off-line robot programming. Apparently, there is a lack of user enthusiasm for standardisation, and perhaps this is primarily due to the fact that industrial users apply robots to perform simple, repetitive, or mass-production tasks which do not require frequent reprogramming. Secondly, major users may well have developed their own interfaces/communication between their CAD system and their robotic devices. However, with the advent of CIM methods this situation is likely to change significantly over the next few decades.

When considering the ‘level’ at which standardisation is likely in the near future, it is concluded that the robot interface data format is the most likely initial target for standardisation with IRDATA and the MMS robot companion standard both offering an important step in the right direction. When internationally agreed robot interface data formats are available, it will be the robot manufacturer who will be responsible for adopting appropriate controller protocols. For commercial reasons, manufacturers have appeared to resist standardisation initiatives and this has resulted in user driven initiatives in the area, such as MAP. As technology advances, new enabling technologies such

as MAP and TOP could provide tremendous capability for information transfer, linking different devices from different vendors, which would truly enhance Computer Integrated Manufacturing and the off-line programming of robots.

7. Conclusions

Post-processors can be system- or application-dependent. Some of these post-processors have been designed for use with an application-dependent robot simulator. The proposed two post-processing methodologies for off-line robot programming, namely the Hierarchical "Top-Down" and "Bottom-Up" approaches which appropriately process data in-line with data structure adopted in CAD model. It would be logical to develop the processor in two modules, pre-processor and post-processor which would substantially improve the efficiency. The successful experience gained proved the validity of the approaches and indicated that similar effort can be applied to different systems. Of course, the total effort will be significantly reduced if standards exist.

References

[1] M. Brady, Robotic technology, Int. J. Robotics Res. 18 (11) (1999) 1051–1055.
[2] J. Owens, Task planning in robot simulation, Ind. Robot 23 (5) (1996) 21–24.
[3] Y.F. Yong, J.A. Gleave, J.L. Green, M.C. Bonney, Off-line programming of robots, Industrial Handbook on Robotics, Wiley, New York, 1985.
[4] G. Gini, M. Gini, Dealing with world model based languages, ACM Trans. Program. Lang. 7 (1985) 334–347.
[5] U. Rembold, R. Dillman, M. Huck, A software system for the simulation of robot based manufacturing process, Faculty for Informatics, Institute for Real-Time and Computer Control and Robotics, Publications of the University of Karlsruhe (TH), Federal Republic of Germany, 1988.
[6] R.K. Stobart, Geometric tools for the off-line programming of robots, Robotica 5 (1987) 273–280.
[7] B.F. Kuvin, Standardized robot cells well prolifically, Weld Design Fabric. 69 (1996) 31–33.
[8] N. Thomson, A common language [manufacturing simulation], Manuf. Eng. 75 (2) (1996) 69–71.
[9] http://www.adept.com/Main/Products/automation/.
[10] J. Dwyer, Simulation—the digital factory, Manuf. Comp. Solut. 5 (3) (1999) 48–50.
[11] B.F. Kuvin, Off-line programming keeps robots working, Weld. Design Fabric. 58 (1985) 34–39.
[12] K. Fernandez, The use of computer graphics simulation in the development of robotic systems, ACTA Astronaut (UK) 17 (1) (1988) 115–122.
[13] C.C. Ruokangas, W.A. Cuthmiller, B.L. Pierson, K.E. Sliwinski, J.M.F. Lee, Off-line programming motion and process commands for robotic welding of space shuttle main engines, J. Robotic Syst. 4 (3) (1987) 355–375.
[14] A. Klein, Off-line programming of painting robots using colour graphics technique, in: Proceedings of the IFIP Conference on Off-line Programming of Industrial Robots, Stuttgart, 1986, pp. 139–151.
[15] P. Grunewald, Car body painting with the spine spray system, in: Proceedings of the 14th International Symposium on Industrial Robots, Gothenburg, 1984, pp. 663–641.
[16] L.B. Frederikson, Robotics in a spray-tip process, in: Proceedings of the 14th International Symposium on Industrial Robots, Gothenburg, 1984, pp. 297–303.
[17] A. Klein, CAD-based off-line programming of painting robots, Robotica 5 (1987) 267–271.
[18] N.A. Yoffa, Off-line programming for automotive spot welding, Robotics World, April 1988, pp. 24–25.
[19] J.A.G. Knight, P.R. Edwards, J. Taylor, To develop a manufacturing system for the production of the decorative patterns on crystal glassware, in: Proceedings of the ACME Research Conference, Science and Engineering Research Council, Salford, 1986.
[20] P.R. Edwards, M. Howarth, Computer integrated crystal class pattern cutting, in: Proceedings of the ACME Research Conference, Science and Engineering Research Council, Nottingham University, 1988.
[21] S.F. Chan, R.H. Weston, K. Case, Robot simulation and off-line programming, Comput.-Aided Eng. J. 5 (4) (1988) 157–162.
[22] A. Watt, 3D Computer Graphics, 3rd ed., Addison-Wesley, Reading, MA, 2000.
[23] U. Rembold, C. Blume, B.J. Frommherz, The proposed robot software interfaces SRL and IRDATA, Robotics Comput. Integrated Manuf. 2 (3/4) (1985) 219–225.
[24] MMS Draft 6 Document, IIEIA project 1393A Draft 6, Manufacturing message specification. Part 1. Service specification, Appendix B: Guidelines for writing Companion Standards, 1987.
[25] L.J. McGuffin, L.O. Reid, R.S. Sparks, MAP/TOP in CIM distributed computing, IEEE Netw. 2 (3) (1988) 23–31.
[26] A. Valenzano, C. Demartini, L. Ciminiera, MAP and TOP Communications—Standards and Applications, Addison-Wesley, Reading, MA, 1992.