An Intelligent Task Programming System for Modular Manufacturing Machines

Proceedings of the Second Singapore International Conference on Intelligent Systems, Singapore, November 14-17, 1994

Dr X.T. Yan[1], Dr K. Case[2] and Professor R.H. Weston[2]

[1] Engineering Design Centre, Engineering Department, Lancaster University, Lancaster LA1 4YR

[2] Department of Manufacturing Engineering, Loughborough University of Technology, Leicestershire LE11 3TU

Abstract

Modular manufacturing machines characterised by their configuration flexibility and low initial investment have increasingly gained more recognition as one of the flexible and responsive manufacturing machinery in current competitive manufacturing industry. Programming of a modular machine is a very important part of an entire machine design and simulation environment. This paper focuses on the programming issue of a modular machine design and simulation environment. A programming study is made of the features of modular machines in manufacturing, and a general systematic approach towards high level utilisation and control of modular machines is outlined. A three-level machine task programming approach within the simulation environment is described and finally an example task program and its execution in simulation environment is illustrated to demonstrate the system capability.

1 Introduction

Modular robots (more generally modular programmable machines) are constructed from modular machine building elements and devices, where both mechanical and control modules can exist within a library of primitives. Such a machine can be configured in a modular manner and the modular elements used can be reused in some other machine building exercise. Modular machines can therefore provide both hardware flexibility and software flexibility, high levels of functionality and cost-effectiveness. They can be designed to meet a given set of application requirements, potentially leading to reduced cycle times, improved accuracy and reduced cost. In addition modular machines can demonstrate sufficient flexibility to (i) automate manufacturing tasks for a range of products and (ii) enable re-configuration as required at some future date, and (iii) most importantly allow them to be truly integrated in a computer integrated manufacturing (CIM) environment [Weston et al 1989].

The increased demand for rapid product change over, equipment utilisation and short lead times leads to a need for a flexible simulation environment, e.g. to maximise the utilisation of automatic machinery [Yong et al. 1988, Siegler et al. 1987, Van Aken and Van Brussel 1988 and Larson and Donath 1985]. To date, both the research literature and commercial modelling and simulation systems focus primarily on robots and means of its off-line programming [Levas and Jayaraman 1989]. However these systems very often do not support various important requirements for modular machine modelling and simulation.

Programming of a modular machine is a very important part of an entire machine design and simulation environment and it provides a user with an interface to communicate with the machine both at simulation and real time control operation stages. The capability of a modular machine is reflected by its programmability and flexibility. This paper focuses on the programming issue of a modular machine design and simulation environment, and its application in real time control. A programming study is made of the features of modular machines in manufacturing, and a general systematic approach towards high level utilisation and control of modular machines is outlined. This approach has been tested on both simulation environment and a physical machine control through the utilisation of a common data format interface between them. A three-level machine task programming environment is described and particular emphasis will be on the intelligence aspect of the task programming. Finally an example task program and its execution in simulation environment is illustrated to demonstrate the system capability.

2 An analysis of modular machine operations

2.1 Generic manufacturing operations

The actions of a manufacturing machine are characterised by two aspects, namely the machine generic operation and the application dependent operations [Volz 1988, Van Aken et al. 1988, Laugier 1988, Sanderson and Homem-de-Mello 1987]. The machine generic operations are defined as those which exist to such a wide extent that they can be found and abstracted in various manufacturing automation machines. The assembly industry provides a typical application for such abstraction of generic operations. Motion, sensory and communication related operations in the context of assembly operations were considered in this paper.

A motion operation in the assembly industry can be further divided into preparatory and task-achieving operations. A preparatory motion is typically used when a component is required to be transferred from its initial position to a position where it is ready to make task achieving operation. A task-achieving motion is typified by a device moving to accomplish a manufacturing task (e.g. insertion, material removal and so forth) by associating the motion with the task related tool.

The signalling operation of sensory devices is another type of generic operation. A sensor generates a signal based on a machine environment change and sends back the signal to the machine controller to make a decision to change the machine state. This is even more important to a manufacturing cell or even a highly automated factory. Therefore, in a computerised manufacturing environment, communication among the devices of a modular machine can also be considered as a generic operation. In associating to sensory devices and communication operation, decision making based on the above operations, is also an important part of intelligent manufacturing processes of automatic machinery, and it is usually determined by the flexibility and intelligence of software programming. This can be generalised into another type of generic operation in intelligent modular machines.

The authors believe that the above four types of generic manufacturing operations encompass most of the generic side of manufacturing machines in assembly and other industries, and that these operations can be used as a target application for modular machine programming.

2.2 Application dependent operations

Each task in different manufacturing applications usually requires application dependent operations to achieve its objectives. Some such special operations are listed as follows:

(1) Spatial operations which change the position of a task related object by non-generic motion;

(2) Geometric operations which change the dimension and shape of a task related object;

(3) Processing one of the physical properties of an object;

(4) Ownership related operations which change the ownership of an object from the previous owner to a new one.

It can be clearly seen that an application dependent operation is characterised by the introduction of a new task description. However this new description can be composed from the four types of generic operations. This can lead to a general approach towards a consistent machine task description with respect to the machine generic operations. There is a need and opportunity to produce a framework for the generic side of the operation together with some sub-set of the application dependent operation descriptions. Thus a robust modular machine programming methodology can be derived.

3 An operation-oriented programming methodology with a three level use1·interface

Based on the above analysis and the modular machine function decomposition, a three level operation oriented machine task description method shown in Figure 1 is proposed and has been implemented to program a modular machine. At the lowest level (machine single primitive level), the task description is focused on the individual primitive feature, and each device is precisely specified to achieve its function in an exact way. This is a level for a complete machine task description. At this level manufacturing operations can be described as *motion generic operations, operation for distributed devices* and *application dependent task.* The task description at the device level provides a better user interface and has some intelligence. Along with the low level machine definition, this level of programming can achieve the machine task description with a reasonable efficiency. A higher level of task description is needed to realise the machine's intelligence, and is denoted as task level programming. Due to space limit only task level programming methods are described in detail as follows. For a complete description of the task programming environment, see Yan [1992].

4 Task level programming

The description of a machine's operation is at a higher level of abstraction and the user can program a modular machine with ease. The premise of this level of programming is that the detailed parameters of each primitive's operation in a device are available to the task program instantiation and decomposition mechanism (PIDM). The task program in the PIDM can reason about some parameter values and primitive operation sequence based on the task level description and available model knowledge and some Macro operation sequences. All these can lead to easy programming with a higher level of abstraction, enhanced intelligence of task program, more rapid program development and intensified program abstraction with automated assistance.

4.1 The abstraction of insertion assembly operation

The geometric information of a modular machine is fully represented in the model and is available to any internal mechanism. The PIDM then has a full spatial knowledge to reason about the distance each motion primitive need to move, whereas the primitive's operation type and the decision of selecting a particular primitive are determined by the high level task description and its incorporated operations. The operation sequence is usually dictated by production rules. For example, the operation sequence for the insertion of a peg into a hole (see Figure 2) may be expressed as

```
MOVE device_name, approaching_postion1;
MOVE device_name, component_gripping_position;
CLOSE gripper_name;
MOVE device_name, approaching_position1;
MOVE device_name, approaching_position2;
MOVE device_name, inserted position;
OPEN gripper_name;
MOVE device_name, approaching_position2;
MOVE device_name, original_position;
```

In order to describe such a task type at a higher level, the user has to specify the device or devices task type, the operation related object names and its destination to uniquely define a task. An implemented task description for the above assembly task is

INSERT *object1_name, object2_name, device_name;*.

The above specification of the object operated upon, operating device and destination object provides all related devices and objects for reasoning and decomposition. The task type INSERT then implies the operation content and sequence at the device level programming. The moving distance for a device is calculated as the relative distance between object1 and object2 along with the assembly rules for INSERT task (in this case the operated component must be approached and inserted normally to its insertion hole with some safety considerations, and this determines the approach positions). Expert systems are expected to be employed for the above purpose and efficient task allocation among several qualified candidate machines in modular machine programming.

4.2 Component spatial information definition and system intelligence

For objects with a simple geometric shape, such as cuboid and cylinder, ten spatial relationships are possible (see Figure 3) excluding the impossible assembly operation of "bottom into" and "beneath". The above ten possibilities are difficult to describe for a user, but can be automatically recognised by an implemented mechanism called intelligent spatial information definition and recogniser (ISIDR). A unit vector is defined for each of these mating holes as an indication of its axis orientation and this information can be accessible to the recogniser mechanism which then deduces the insertion direction based on the unit vector definition of two mating objects. If a machine's task is to place a component onto an object, it can work out the placing direction according to the unit vector of the target location. This ISIDR mechanism has greatly help the higher level task program to be deduced into a number of events through decomposition and the intelligence of the programming system is clearly illustrated.

4.3 The task programming instantiation and decomposition mechanism

Knowledge about the two mating components gives more information for the system to calculate the moving distance. It is easy to calculate the distance between the two frames related to the component local frame. Through a search of the modelling data structures of two component geometry, the boundary, shape and dimensions of each component can be found, and the offset between the coordinate frame and the outline of each component can then be calculated.

Since a set of conventions are used to specify the position and orientation of a local coordinate frame relative to the geometric representation in GRASP [Glib 1989] and other equivalent modelling systems, the calculation of a coordinate frame offset can be used to obtain the moving distance due to the component dimension variation. The spatial relationship between two mating components is also a very useful in calculation of the moving distance. The distance due to the assembly direction can be obtained on the spatial information specified in the object level task programming. With the above full knowledge and the names of the devices involved, a lower level (device level) of machine task execution programming can be generated with reference to the task description and the assembly rule.

The object level programming described partially achieves the objective of intelligent programming. A genuine intelligent programming approach needs to be dealt with at an even higher level - assembly task programming, and this approach requires more background information and high level intelligent reasoning. However, this area remains untouched and is left for future research.

5 Conclusion

This paper has illustrated some implementation of modular machine programming which is essential part of the modular machine design and simulation environment. An analysis of manufacturing operation has been conducted and they have been categorised into generic and application dependent types. A three level modular machine programming environment is described and their implementation has been discussed with particular reference to assembly operations. Articulated and distributed modular machine configurations are considered for their programming. The system incorporates artificial intelligence to some extent and object oriented programming is desirable for generating high level task programming.

References:

Glib, "Grasp Library Subroutine Manual", BYG System Limited, Nottingham, 1989.

Larson, R. and Donath, M., "Animated Simulation of Intelligent Robot Workcell", Proceedings of Robots 9 SME Ref. MS85-622, pp. 19-54-19-69, Detroit, Michigan, USA, June 2-6, 1985.

Laugier, C., "Planning robot motions in the SHARP system," in "CAD Based Programming for Sensory Robots", Edited by Bahram Ravani, Published by Springer-Verlag Berlin Heidelberg, NOTO ASI Series, Vol. F50, pp.151-187, 1988.

Levas, A. and Jayaraman, R, "WADE: An Object-Oriented Enviromnent for Modelling and simulation of Workcell Applications," IEEE Transactions on Robotics and Automation, Vol. 5, No 3, pp. 324-335, June 1989.

Sanderson, A.C. and Homem-de-Mello, L.S., "Task Planning and Control Synthesis for Flexible Assembly Systems", in "Machine Intelligence and Knowledge Engineering for Robotic Applications", Edited by Wong, A.K.C. and Pugh, A., NATO ASI Series Vol. F33, Springer-Verlag, pp. 331-353, 1987.

Siegler, A., Bathor, M. and Devi, G., "A 3-Dimensional Computer Animation System with Robotic Applications", Robotica, Vol. 5, pp. 281-290, 1987.

Van Aken, L. and Van Brussel, H., "Robot Programming Languages: the Statement of a Problem", Robotica, Vol. 6, pp. 141-148, 1988.

Van Aken, L., Van Brussel, H. and Schutter, J.D., "Simplification of a Robot Task specification by Incorporating a Structured Geometric Database into an Off-line Robot Programming System", CAD Based Programming for Sensory Robots, Edited by Ravani, B., NATO ASI Series F, Vol. 50, pp.123-150, July 4-6, 1988.

Volz, R.A., "Report of the Robot Programming Language Working Group: NATO Workshop on Robot Programming Languages," IEEE Journal of Robotics and Automation, Vol. 4, No. 1, pp. 86-90, February 1988.

Weston, R.H., Harrison, R., Booth, A.H., and Moore, P.R., "Universal Machine Control System Primitives for Modular Distributed Manipulator Systems," International Journal of Production Research, Vol. 27, No.3, pp. 393-410, 1989.

Yan, X.T., "Graphical modelling of modular machines", Ph.D. Thesis, Department of Manufacturing Engineering, Loughborough University, 1992.

Yong, Y.F., Marshall, R.J. and Bonney, M. C., "Using CAD to Plan and Evaluate a Robotic Cell", in Control and Programming in Advanced Manufacturing ". Edited by Rathmill, K., Published by IFS Ltd, UK, Springer-Verlag, pp. 235-247, 1988.

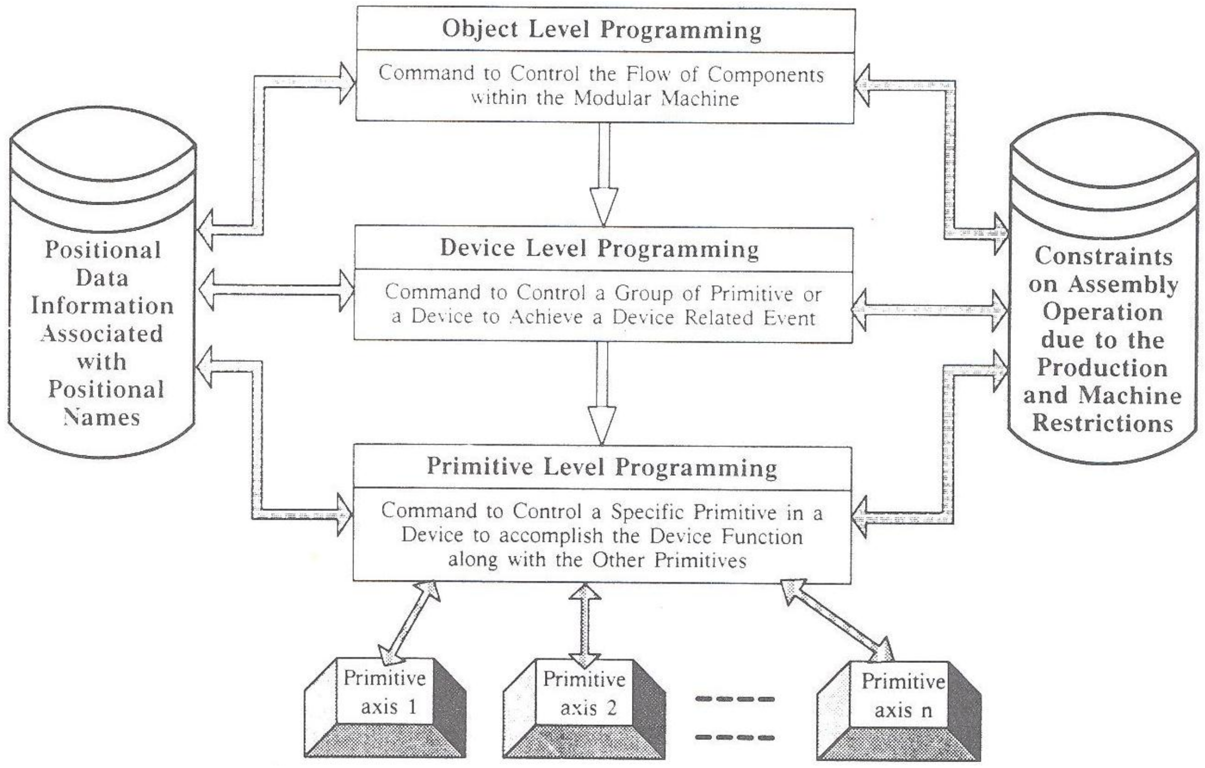
Object Level Programming
Command to Control the Flow of Components within the Modular Machine
Device Level Programming
Command to Control a Group of Primitive or a Device to Achieve a Device Related Event
Primitive Level Programming
Command to Control a Specific Primitive in a Device to accomplish the Device Function along with the Other Primitives
Positional Data Information Associated with Positional Names
Constraints on Assembly Operation due to the Production and Machine Restrictions
Primitive axis 1
Primitive axis 2
Primitive axis n

Figure 1 A three level task programming system for modular manufacturing machines

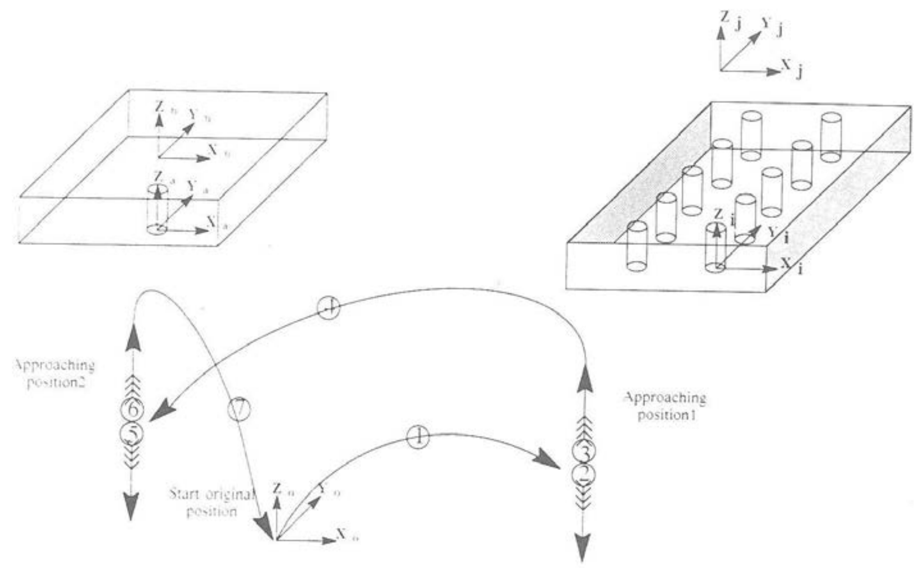

Figure 2 A typical insertion operation procedure

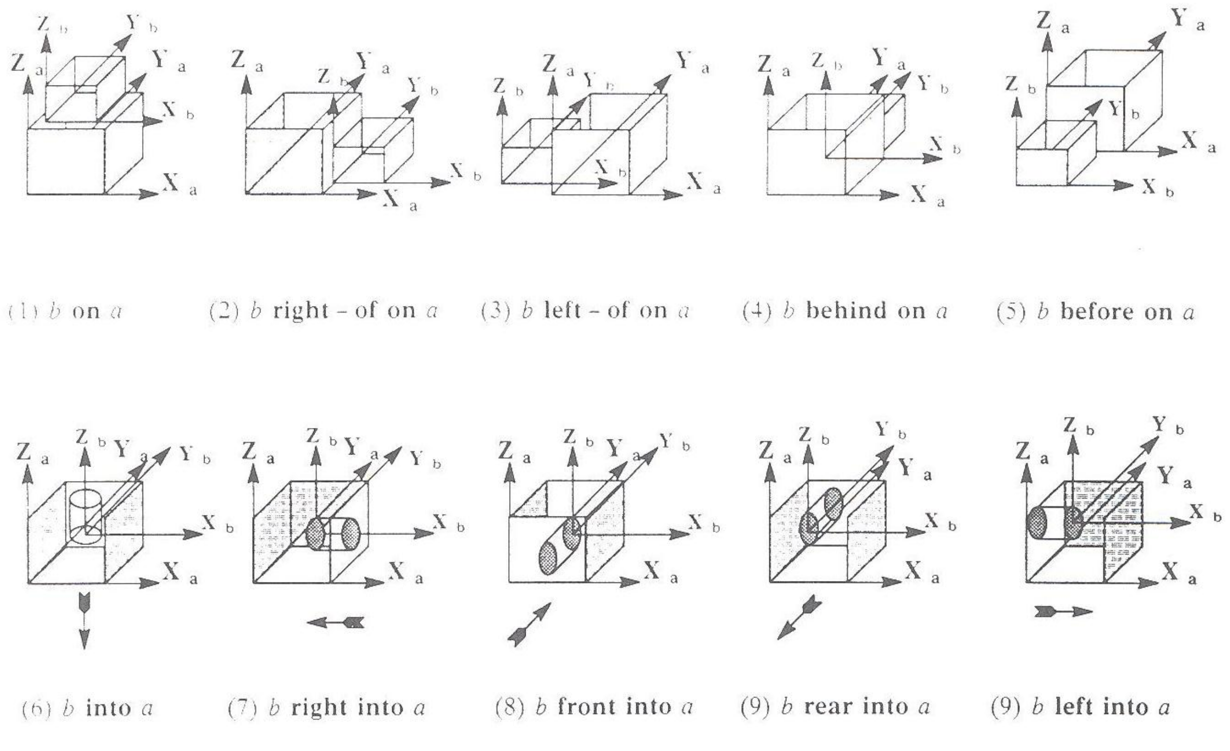

Figure 3 Ten possible relationships of insertion and placing operation between two objects