# THE INDUSTRIAL ROBOT






The Industrial Robot is published quarterly by International Fluidics Services Ltd., 35-39 High Street, Kempston, Bedford MK42 7BT, England.
(Tel: Bedford (0234) 853605)
(Telex: 825489)

The journal is available on Subscription at £28.00 UK subscribers US$65.00 elsewhere (Overseas) per volume





The Industrial Robot is printed by The Cotswold Press, Oxford.

The Industrial Robot technical articles are indexed in Engineering Index, USA.

# Object description with a manipulator

G.Gini and M.Gini, Polytecnico di Milano, Italy


## ABSTRACT

This paper presents an interactive system, called POINTY. The goal of this system is to solve problems encountered when a high-level manipulator lanaguage, as AL, is employed.

Those problems are related to the necessity of supplying any manipulation program with a complete description of the objects involved. The object description has an important role in any task description oriented language. In these languages the procedural part, which expresses the assembly steps, is reduced by increasing correspondently the descriptive part, which expresses the knowledge a program has about the physical world.

An approach to the generation of object models is presented, and the system based on it is illustrated. The basic idea is to point to the objects with the manipulator for building an incremental model of the world, and then to generate the AL corresponding instructions.

The preliminary experiences using it demonstrate that the object model part of an AL program can be easily obtained and tested.


## 1. INTRODUCTION

The problems encountered when a task description programming language is used in programming an automatic manipulator are discussed. A system to generate a complete description of the world in a task oriented manipulator language is presented. This system interprets a subset of AL, a high level, Algol-like language for automation, interacts with the manipulator to build an incremental model of the world, and then constructs the AL statements corresponding to it.

Computer controlled manipulators have emerged as an important class of machines in recent years. Until recently little attention has been paid to the development of programming languages for such applications as arm control and automatic assembly. The only programming method in common industrial use today is the "teach mode"; the user does not write a program, but moves the arm around the work station to define all the assembly steps. Most research laboratories and companies have chosen to pursue this technique instead of developing high-level languages.

The greater generality offered by a system based on a formal programming language suggested the definition of special languages for assembly task descriptions. In this direction few languages were developed; AL, a high level programming language for automation, implemented at the Stanford Artificial Intelligence Laboratory [1], is the first complete achievement. In any AL program the initial locations of the objects and the relationships between them and their subparts have to be precisely defined and then all the assembly steps are expressed in terms of those definitions. This object description is longer and more difficult to code than the procedural statements expressing the movemants, and the complexity of real industrial parts vastly increases the likelihood of programmer errors and the time for accurate coding.

Since the motivation behind high level manipulator languages is the ease of writing programs, it seems advisable to automate the generation of the desired world model. For this purpose the system presented here, POINTY, has been developed and implemented [2].

Its basic idea is to use the manipulator as a measuring tool in three dimensions to point to the objects while simple AL command are interpreted, for building an internal incremental model of the world, and then for generating the AL corresponding description.

Besides the availability of computer controlled movements of the arm allows interpreting parts of AL programs and checking the correctness of them and of the model being defined.

The idea of interacting with the external world to generate the world model is an important aspect for further developments of manipulation languages, aimed at reducing more and more the complexity of programming, mainly for non expert users. High level systems to build the world model description, to check step by step the execution of the program, and to construct parts of programs could be usefully developed for training in industrial applications.

## 2. USING A TASK DESCRIPTION LANGUAGE

The purpose of any high-level manipulator programming system is to provide the user with effective means of specifying what he wants done by the manipulator to achieve a desired task, without having to be worried about a lot of details necessary to control the manipulator but not very related to the assembly operations.

The most natural description of an assembly task is in terms of the desired effects on the parts being assembled, rather than in terms of the manipulator movements. The description is so an ordered list of assembly steps like grasp an object, move the object to a given position, ungrasp it and so no. It is quite obvious to observe that some specifications of the objects involved in the operations have to be supplied, like their position, the geometrical shape or some other features.

A system which transforms that high-level description of mechanical assembly operations into a program for execution

by a computer controlled manipulator seems to be an important requirement to ease the complex task of writing assembly programs.

The problem is how to obtain from the description of the objects, which we shall call world model, and the description of the assembly steps a manipulator level program, in which the functional capabilities of the manipulator are applied to perform the required task.

To fill up the gap between those two levels many decisions have to be taken by the system and many problems solved. Most of the problems are at the planning and representation level, because the task description does not define in a complete and unambiguous way the required operations and does not supply enough information about the control of the arm movements, the speed and the acceleration to use, the forces to exert, the trajectories required [3].

So far, a system in which the user describes what he wants done and the computer writes the corresponding manipulator program has not yet been implemented; the solution of open research problems in artificial intelligence and manipulation would provide the basis for this new generation of manipulator languages [4]. However few languages in that direction, AL [1], AUTOPASS [5], LAMA [6] have been implemented in recent years.

In this presentation, we shall refer to the experimental system for programmed assembly designed and developed at Stanford Aritificial Intelligence Laboratory [7]. The system is mainly composed by two Stanford Scheiman arms [8], with six degrees of freedom, which allow them to be positioned at any arbitrary position and in any arbitrary orientation, and software compiled by a PDP 10 and running on a devoted PDP 11/45. The programming language here developed, AL [1] [9], allows the user a quite natural and synthetic task level description of assembly sequences, while the world model required is more complex.

It is important to realize that the complexity of that world description required by AL programs is a direct consequence of the way chosen to solve the representation and planning problems before outlined. AL tends to reduce the complexity of the description of the assembly steps, increasing correspondently the description of the parts involved in the assembly. Before showing the amount of the effort required to the user to write such a description we shall summarize some of the AL features.

AL provides different data types, Algol-like control structures, arithmetic operations and movement instructions for describing the physical entities and the manipulation steps. To describe manipulator positions, object locations and their subparts AL uses frames, which represent a co-ordinate system.

The manipulation task usually requires to have several frames associated with the same object, each one representing an important aspect, for instance the grasping position, or a reference point, or another feature of the object. When the object is moved during the assembly, all the frames associated assume new values. It would be long to change all these values explicitly. The affix construct of AL allows the user to specify that a variable will be computed from other variables. So, when an object is assembled with another, or when it is grasped by the manipulator, we write in the program, for instance, "AFFIX BOX TO BARM;", where BARM is the name of the arm involved. A change in the position of BARM or BOX will automatically update the value of the affixed frame.

The variables, their velues and the data structures associated with affixments form the AL model of the world. The world model is a tree of affixed frames: the nodes represent the physical objects or subparts of them, and the arcs the relationships between them. The root of the tree is an implicit object called WORLD, and all objects which are not subparts of anything else are subparts of WORLD. Each arc also contains information concerning the relative transformations between the two nodes and specifies whether the relationship is rigid, non-rigid or independent.

The motion steps are very easy to code using that world model. Since the purpose of any manipulation task is to move objects rather than to get the manipulator in some defined position and orientation, AL allows to describe motions in terms of frames. For instance, if we want to move a box to a desired final position, we write the AL instructions:

```
AFFIX BOX TO BARM;
MOVE BOX TO FIN_POS;
```

and changes in the value of BARM will cause corresponding changes in the value of BOX. The value of FIN_POS is then used to produce the hand position that will cause BOX to be at the final place.

All the motion statements may be enriched by adding a lot of modifying clauses, for example via points, approach and deproach positions, controls on force sensors, which allow the manipulator program obtained to be able to perform sophisticated tasks.

A simple example can better show the world model by AL programs; the scene illustrated in Fig.1 is described by the following AL instructions.

```
FRAME BEAM, BEAM_HOLE;
BEAM ← FRAME (NOLROTN, VECTOR(40,60,0));
AFFIX BEAM_HOLE TO BEAM RIGIDLY AT TRANS
  (ROT(YHAT,90), VECTOR(0,1.5,6.0);

FRAME BRACKET, BRACKET_GRASP, BRACKET_
  HOLE;

BRACKET ← FRAME (NILROTN,VECTOR(10,40,0));
AFFIX BRACKET_HOLE TO BRACKET RIGIDLY AT
  TRANS (ROT(YHAT,180), VECTOR(5.1,2,0));
AFFIX BRACKET_GRASP TO BRACKET RIGIDLY AT
  TRANS (ROT(YHAT,180), VECTOR(5.1,0,5));

FRAME BOLT;
BOLT ← FRAME (ROT(YHAT,180), VECTOR(30,20,5));
```

A schematic description of the model, in which all the parts are arranged hierarchically are represented as nodes in the frame tree is shown in Fig.2.

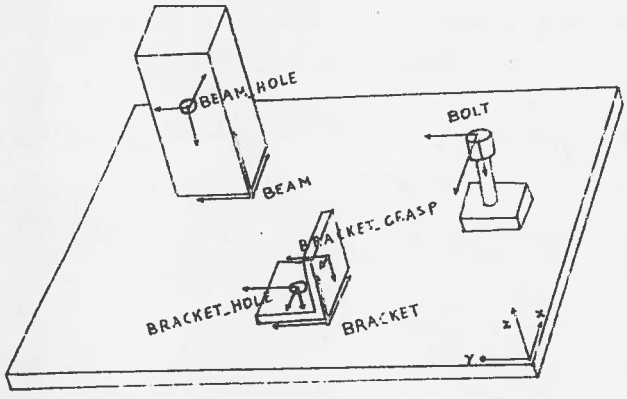


Fig.1

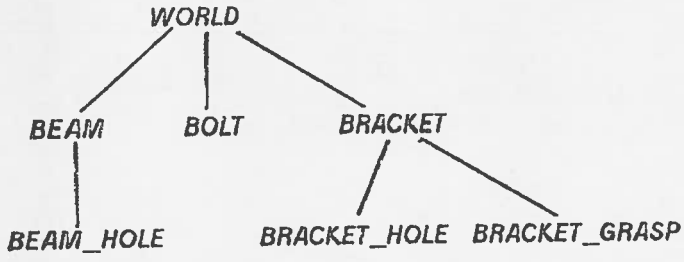


Fig.2

Next section will be devoted to the problem of easing the world model construction.

## 3. TOWARDS THE AUTOMATIC GENERATION OF OBJECT MODELS

This paper proposes a prototype system which would semi-automate the process of specifying object models for AL programs.

The complete automation of this part would be very complex: most of the robots are blind or, when there is a TV camera, the main problem is how to extract from the picture only the necessary information, without having to be concerned with a big mass of data. Such a problem has no adequate solution at the moment, and we shall propose a different approach.

The idea developed here is to try to supply the user with some simple means to describe his object model. The decision about what are the important features for the assembly process, what is the best way to define the objects is totally given to the user, while the reading of the arm location, the definition of the object positions, the computation of the transformations between different reference systems and the generation of the corresponding AL instructions are given to the system. In that way the user can concentrate his effort in the most interesting aspects.

A system for interactive modelling, POINTY, has been designed and implemented at Stanford Artificial Intelligence Laboratory [2].

Pointing, which is the method first proposed in [10] and developed in POINTY, involves an interactive system in which the data structures are built by using the manipulator and a special tool to point to the objects. The manipulator may be moved around manually, or under computer control. It is possible to infer the position of the pointed object from the known joint angles of the manipulator and from the relative position of the pointer with respect to the arm. A sequence of AL instructions corresponding to the defined world model, to be used in an AL program, can then be produced.

The language recognized by POINTY is the same as AL, although some special instructions have been added to handle the pointing method. In that way a very high-level language is provided, in which a single instruction performs complex operations like reading the position of the pointer and assigning its value to a frame, defining affixments, and operating on the frame tree. In addition a lot of user facilities, as error recovering and editing, error messages, display of the model on the screen, interaction with disk files are available.

The choice of having the same input language for the two parts of the manipulation program, the object description and the task description, has been motivated by considerations such as uniformity in program formulation, ease in documentation, naturalness in integrating the two descriptions. Besides the use of the same implementation language used for AL, SAIL [11], allows sharing modules and internal structures.

The instruction set can be divided in the following classes.

**operations on variables** are mainly declarations, assignment of values, arithmetic operations and deletions. Values can be absolute or relative to another reference system and arithmetic operations as in AL are allowed. Declarations can be avoided, deletions have been introduced to improve the interactive use.

**operations on tree structure** allow to build and modify the tree structure which constitutes the internal model representation. Affixment, unfixment, copy and merge are some examples of these instructions.

**interactions with the manipulator** are used to move the arms and to read their position. Often the objects are pointed by a special pointer and it is possible to infer the position of the object from the arm position and from the relative transformation between the arm and the pointer position. Since it can be quite difficult to guide manually the manipulator to a desired orientation, the definition of frames with some widely used orientations can be easily obtained. The availability of computer controlled movements of the arm allows checking the correctness of the model being defined.

**input/output operations on files** allow to read files with AL instructions and to output the AL instructions corresponding to the defined model.

system facilities are mainly editing of variable values, renaming,

some error recovering procedures, information on the syntactic errors, general information on the system. Besides abbreviated forms, default values and the possibility of killing the last instruction and its side effects are provided.

Some of the features of the system can be illustrated by using the previous example. It is easy to see that the program is mostly a request to read the pointer position and to assign its value with a desired orientation part to the indicated frame. No computations about relative transformations have to be done by the user. A simple "write" instruction generates the AL instructions corresponding to the defined model and outputs them on a file.

```
BEAM ← ↑ INPUT
BEAM_HOLE ← (0,90,0,0,1.5,6) REL BEAM
AFFIX BEAM_HOLE TO BEAM

BRACKET ← ↑INPUT
BRACKET_ HOLE← ↓ INPUT
AFFIX BRACKET_HOLE TO BRACKET
BRACKET_GRASP← ↓ INPUT
AFFIX BRACKET_GRASP TO BRACKET

BOLT← ↓ INPUT

WRITE
```

## 4. CONCLUSIONS AND EXTENSIONS

POINTY, an interactive system that allows the user to interactively build data structures of AL programs using the manipulator itself to point to the objects, has been presented. The implementation has been tested, demonstrating that the specification of object models may be easily obtained.

The aim of this system is to be a tool for the world model definition related to the current version of AL. Many future developments can be added following improvements in the manipulation language; for example the definition of the geometric shape of objects now is not used by AL, but may be an important part in a more advanced system. New instructions may be easily added to POINTY to increase the facilities offered by the system and to extend the compatibility with AL.

Most of the proposed system could be easily adapted for other methods of pointing, for other high level manipulation languages or other arms.

The idea of generating not only the world model but also the sequence of instructions required to execute the assembly process could be considered.

An interactive system providing world model and program generation could be usefully developed for training in industrial applications. Such a system would provide an easy way to define and check a program in a high level interactive language, and then to output the complete program, for the compilation and reiterated executions.

## 5. REFERENCES


1. Finkel,R., Taylor,R., Bolles,R., Paul,R., and Feldman,J. "AL, a programming system for automation", Stanford Artificial Intelligence Laboratory Memo AIM-243, Stanford Computer Science Report STAN-CS-74-456, Stanford, Ca. (November 1974).

2. Gini,C. and Gini,M. "Interactive modelling for AL", in T.O.Binford et al: Exploratory study of computer integrated assembly systems, Stanford Artificial Intelligence Laboratory, Memo AIM-285.4, Stanford Computer Science Report STAN-CS-76-568, Progress Report 4 (1 August 76/31 March 77).

3. Taylor,R.H. "A synthesis of manipulator control programs from task-level specifications", Stanford Artificial Laboratory Memo AIM-282, Stanford Computer Science Report STAN-CS-76-560, Stanford, Ca. (July 1976).

4. Fahlman,S.E. "A planning system for robot construction tasks". MIT Artificial Intelligence Laboratory, Tech Report 283, MIT, Cambridge, Mass. (May 1973).

5. Lieberman,L.I. and Wesley,M.A. "AUTOPASS, a very high level programming language for mechanical assembler systems", IBM Research Report RC-5599 (August 1975).

6. Lozano-Perez,T. "The design of a mechanical assembly system", MIT Artificial Intelligence Laboratory, Tech Report 397, MIT, Cambridge, Mass. (December 1976).

7. Binford,T.O. et al. "Exploratory study of computer integrated assembly systems", Stanford Artificial Intelligence Laboratory Mem AIM-285, Stanford Computer Science Report STAN-CS-76-568, Stanford, Ca. (August 1976).

8. Scheinman,V.D. "Design of a computer controlled manipulator", Stanford Artificial Intelligence Project, Memo AIM-92. (June 1969).

9. Finkel,R., Taylor,R., Bolles,R., Paul,R. and Feldman,J. "An overview of AL, a programming system for automation", Proc. 4th International Joint Conference on Artificial Intelligence, pp.758-765, Tbilisi. (September 1975).

10. Grossman,D.D. and Taylor,R.H. "Interactive generation of object models with a manipulator", Stanford Artificial Intelligence Laboratory, Memo AIM-274, Stanford Computer Science Report STAN-CS-75-536, Stanford, Ca. (December 1975).

11. Reiser,J.R. editor "SAIL", Stanford Artificial Intelligence Laboratory Memo AIM-289, Stanford Computer Science Report STAN-CS-76-574, Stanford,Ca. (August 1976).