Proceedings

8th International Symposium On Industrial Robots

10. IPA ARBEITSTAGUNG

4th International Conference On Industrial Robot Technology

Stuttgart, West Germany
30. 5. 1978 — 1. 6. 1978

VERANSTALTER/ORGANISER
Institut für Produktionstechnik
und Automatisierung (IPA), Stuttgart
in Zusammenarbeit mit:
International Fluidics Services Ltd.,
Kempston, Bedford, England

UNTERSTÜTZT DURCH/SUPPORTED BY
REFA, Darmstadt
VDMA, Frankfurt
RKW, Landesgruppe Baden-Württemberg
VDI Gesellschaft Produktionstechnik
IFToMM, Warschau, Polen

AUTHOR INDEX / REFERENTEN VERZEICHNIS

USING A TASK DESCRIPTION LANGUAGE FOR ASSEMBLY. THE GENERATION OF WORLD MODELS

by Giuseppina Gini° and Maria Gini °

Artificial Intelligence Laboratory
Stanford University, Stanford, California

SUMMARY

This paper intends to introduce a system for constructing and using world models in languages for computer controlled manipulators.

The basic idea of high-level manipulator languages is to allow the description of the task to be carried out in a task oriented way. The possibility of expressing tasks by a set of high-level operations is strictly related to the availability of a world model. Only if the program has some knowledge about the external world, the objects and the relations between parts and subparts, the motion statements can be expressed in terms of the objects, and not in terms of manipulator movements.

In those languages the procedural part, which expresses the assembly steps, is reduced to a simple sequence of high-level instructions, by increasing correspondently the knowledge that the program must have about the physical world.

An approach to the interactive generation of object models is presented, and a working system based on it, POINTY, is illustrated. The basic idea is to point to the objects with the manipulator itself, to build an incremental model of the world, and then to generate the corresponding descriptions in the high-level manipulator language.

The proposed system is part of the AL system, a software system for programmable automation developed at Stanford Artificial Intelligence Laboratory.

VERWENDUNG EINER AUFGABENBESCHREIBUNGSSPRACHE FÜR DIE MONTAGE

Es wird über ein System zum Bau von Computer gesteuerten Manipulatoren bei Verwendung von Weltmodellsprachen referiert.

Die Grundidee, Manipulatorsprachen von hohem Niveau zu entwickeln, bezweckt, daß die Beschreibung einer Aufgabe aufgabenorieniert ausgeführt wird. Die Möglichkeit, Aufgaben durch einen Satz von Operationen hohen Niveaus auszudrücken, hängt sehr davon ab, ob ein Weltmodell verfügbar ist. Nur wenn das Programm über Kenntnisse der externen Welt, der Objekte und der Beziehungen zwischen Teilen und Unterteilen verfügt, lassen sich die Bewegungssätze objektorientiert und nicht bewegungsorientiert ausdrücken.

In diesen Sprachen wird der Prozedurteil, der die Montageschritte ausdrückt, auf eine einfache Sequenz von Befehlen hohen Niveaus beschränkt, indem die Kenntnisse, die das Programm über die physikalische Welt besitzen muß, erheblich erweitert werden.

Es wird ein Versuch, die Objektmodelle interaktiv zu erzeugen, vorgestellt und es wird ein darauf basierendes Arbeitssystem, POINTY genannt, geschildert. Die Grundidee ist, mit einem Manipulator auf die Objekte hinzudeuten, um ein differentielles Modell der Welt aufzubauen und erst dann die entsprechenden Beschreibungen in der Manipulatorsprache hohen Niveaus zu erzeugen.

Das vorgeschlagene System ist ein Teil des AL-Systems, eines Softwaresystems zur programmierbaren Automation, das im Stanforder Laboratorium für künstliche Intelligenz entwickelt wurde.

° Present affiliation:
Istituto di Elettrotecnica ed Elettronica
Politecnico di Milano, Italy

I - INTRODUCTION

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 strictly 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 that object, move it to a given position, ungrasp it and so on. 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. The most of the problems are at the planning and representation level, because the task description doesn't define in a complete and unambigous way the required operations and doesn't supply enough information about the control of the arm movements, the speed and the acceleration to use, the forces to exert (Ref. 1).

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 manipulation would provide the basis for this new generation of manipulator languages. However few languages in that direction, AL (Ref. 2, 3), AUTOPASS (Ref. 4), LAMA (Ref. 5) have been proposed or implemented in recent years (Ref. 6).

In this presentation, we shall refer to the experimental system for programmable assembly designed and developed at Stanford Artificial Intelligence Laboratory (Ref. 7, 8). The system is mainly composed by two Stanford Scheiman arms (Ref. 9), 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, allows the user a quite natural and synthetic task level description of assembly sequences, based on a precise description of the world model.

We intend to discuss the relevance of using a world model description in order to have a more natural and clear way to describe assembly tasks. In consideration of the fact that describing world models can be tedious and time spending, we will show in this paper how is possible to obtain world model descriptions in an interactive way, and how these descriptions can be written in the same high-level language used for coding the assembly steps.

II - USING A TASK DESCRIPTION LANGUAGE FOR ASSEMBLY

To allow the user to describe his assembly job with simple instructions like move handle to handle_position, move pin to handle_hole ..., the program must exactly know what handle, handle_position, pin, handle_hole are. That means the program must operate on an internal model of the world, which it can always refer to.

It is important to realize that the complexity (in terms of number of instructions and accuracy of numerical values) of that world description required by programs written in a language as AL 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 efforts 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, coordination primitives, 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 coordinate system in the cartesian space.

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 must 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. 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 RIGIDLY;", 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 values 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 the 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, nonrigid or independent.

The motion steps are very easily coded 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 RIGIDLY;
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.

III - WHY A WORLD MODEL IN MANIPULATOR LANGUAGES

A simple example can better show the world model used by AL programs; the scene illustrated in Fig. 3.1 is described by the following AL instructions.

```
FRAME ORIGIN, STEM, STEM_GRASP, NUT, BASE, BASE_STEM, BASE_NUT;
ORIGIN ← FRAME (NILROTN, VECTOR (5.86, 25.7, .290) * INCHES);
AFFIX STEM TO ORIGIN AT
     TRANS (NILROTN, VECTOR (10.5, 2.00, .0) * INCHES) NONRIGIDLY;
AFFIX STEM_GRASP TO STEM AT
     TRANS (ROT (YHAT, 111.1533)* ROT (ZHAT, -3.6397),
            VECTOR (-.457, -.135, 1.87) * INCHES) RIGIDLY;
AFFIX NUT TO ORIGIN AT
     TRANS (NILROTN, VECTOR (14.5, 10.0, .0) * INCHES) NONRIGIDLY;
AFFIX BASE TO ORIGIN AT
     TRANS (NILROTN, VECTOR (2.60, 2.00, .0) * INCHES) NONRIGIDLY;
AFFIX BASE_STEM TO BASE AT
     TRANS (NILROTN, VECTOR (.150, .0, 2.16) * INCHES) RIGIDLY;
AFFIX BASE_NUT TO BASE AT
     TRANS (NILROTN, VECTOR (1.60, .0, 2.40) * INCHFS) RIGIDLY;
```

where the specification rigidly means that the affixment relation is symmetric, while nonrigidly means a one-directional relation.

A schematic description of the model, in which all the parts are hierarchically arranged and represented as nodes in the frame tree is shown in Fig. 3.2. The arcs in the tree are marked according to the type of the affixment relation.

The advantages for the user in having a system able to manipulate world models are pretty obvious.

First, the possibility of using high-level manipulation instructions referred to the objects or their positions preserves part of the semantic content that the user puts in his program. It is important to observe that the knowledge about the workpieces and the assembly steps is generally obscurated by the manipulator programs in which the objects are not represented but only the manipulator positions are used.

Second, the same assembly program can be used for different configurations of the parts in the assembly station. In this case we do not need to change the assembly sequence but only to reassign the new values to the objects. The affix instruction takes care of the book-keeping. As illustrated in the example, it is a good practice to define the positions of the objects as relative to a reference point, a corner of the fixture generally. So a translation or a rotation of the fixture in the work station requires only the appropriate modifications of that reference point.

Another important aspect is the possibility of using that world model to compute at compile time the arm trajectories or to implement a collision avoidance system.

In next section we will examine how to generate world models corresponding to that frame tree, vectors and variables related, and how to generate the corresponding AL code.

IV - THE GENERATION OF WORLD MODELS

Generating world model descriptions of a given scene is an ambitious task requiring a lot of geometric and physical knowledge and the solution of many world representation problems. Our approach to the construction of such descriptions is quite different, because we don't intend to generate automatically world models from vision systems, but only generate them automatically from an internal model, interactively constructed by pointing the arm to the desired positions and orientations.

This approach is quite similar to the classical teaching by doing mode, as effort required to the user, but differs deeply as result. In teaching by doing sessions we obtain some positions to be used in a program, but we don't obtain world models. So, we don't have any possibility to use the work done if we change in some way the assembly station. Having a world model allows programs to be simpler and more readable, and allows parametric definitions, useful for different situations.

The idea developed here is to supply the user with some simple means to describe his object models. 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 trasformations between different reference systems and the generation of the corresponding AL instructions are given to the system.

A system for interactive modelling, POINTY (Ref. 3, 8), has been proposed and implemented at Stanford Artificial Intelligence Laboratory.

Pointing, which is the method first proposed in (Ref. 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 be then automatically 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 such considerations 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 (Ref. 11), allows sharing modules and internal structures.

The instruction set of POINTY can be diveded in

(i) Operations on the world model.
Are mainly variable declarations, assignment of values, arithmetic operations and deletions. Values can be absolute or relative to another reference system and arithmetic expressions are recognized. Declarations can be avoided. 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.

(ii) Interactions with the manipulator.
Are used to move the arms and to read their positions. 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.

(iii) User interface.
Input/output operations on files allow to read files with AL instructions and to output the AL instructions corresponding to the defined world model. Editing of variable values, renaming, some error recovering procedures, information on the syntactic errors, and general information on the system are available. Besides abbreviated forms, default values and the possibility of killing the last instruction and its side effects are provided. A display is generated and automatically updated on the screen with the state of the world model being defined.

The characteristics and the use of the system can be illustrated by describing how to program the previously given example.

It's easy to see that the program is mostly a request to assign values read on the arm 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.

First we observe that it's better to start using the pointer and defining all the positions not accessible by the hand. We can calibrate the pointer everytime, but for the standard pointer we can use the standard calibration provided by the system.

We start defining, as reference point, a frame located at one corner of the main fixture

```
ORIGIN ← ↑ INPUT
```

The up arrow indicates that we want ORIGIN to have the location at the end of pointer, but with WORLD orientation.

```
AFFIX BASE TO ORIGIN AT TRANS (NILROTN, (2.6, 2, 0)) NONRIGIDLY
```

This instruction defines a new frame, BASE, using the given transformation as relative position of BASE to ORIGIN and affixes the new frame to ORIGIN. NONRIGIDLY can be abbreviated by +. In the same way we can define other frames

```
AFFIX STEM TO ORIGIN AT TRANS (NILROTN, (10.5, 2, 0)) +
BASE_STEM ← ↑ INPUT
AFFIX BASE_STEM TO BASE
```

Now we don't need anymore the pointer and we can remove it; we continue the session using the arm.

```
STEM_GRASP ← INPUT BARM
AFFIX STEM_GRASP TO STEM
NUT ← ↑ INPUT BARM
AFFIX NUT TO ORIGIN +
```

We now move BARM with NUT between the fingers to the final position and then we type

```
BASE_NUT ← ↑ INPUT BARM
AFFIX BASE_NUT TO BASE
WRITE
EXIT
```

In Fig. 4.1 the world model defined is represented as displayed on the screen.

V - CONCLUSIONS

We introduced some considerations about the opportunity of having high-level manipulator languages, based on world models.

After introducing AL, one of such languages, we presented an interactive system designed for providing AL world models by interacting with the physical world.

We demonstrated that such a system can be usefully used in order to simplify the writing of AL programs.

Let us now show what a system like POINTY can do more. The idea of describing the world using the same language as for describing assembly programs, means that we have a language complete enough for both the purposes. The implementation of POINTY so is not only the implementation of an interactive module, but in some sense is an interpreter of a high-level assembly language. The possibility of producing complete AL programs as result of POINTY sessions could be investigated and developed. In that way POINTY would become a system able to generate AL programs from examples of assembly.

These ideas will be developed in a short time; we mentioned them in order to show a more global view of the system.

ACKNOWLEDGEMENTS

We would like to express our thanks to Tom Binford for discussions, suggestions and comments, to all the other members of the Hand-Eye group for the long cooperation, and in particular to Shahid Mujtaba for his continous help in implementation.

REFERENCES

1. Taylor, R.H. "A synthesis of manipulator control programs from task-level specifications", Stanford Artificial Intelligence Laboratory Memo AIM-282, Stanford, CA. (July, 1976).

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

3. Mujtaba, M.S., Goldman, A. "AL users' manual", Stanford Artificial Intelligence Laboratory, Stanford, CA. (November, 1977).

4. Lieberman, L.I., Wesley, M.A. "AUTOPASS: an automatic programming system for computer controlled mechanical assembly", IBM Journal of Research and Development. (July, 1977).

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

6. Park, W.T. "Minicomputer software organization for control of industrial Robots", 1977 Joint Automatic Control Conference, San Francisco, CA.

(June, 1977).

7. Binford, T.O., Grossman, D.D., Liu, C.R., Bolles, R.C., Finkel, R.A., Mujtaba, M.S., Roderick, M.D., Shimano, B.E., Taylor, R.H., Goldman, R. H., Jarvis, J.P., Scheinman, V.D., Gafford, T.A. "Exploratory study of computer integrated assembly systems", Progress Report 3, Stanford Artificial Intelligence Laboratory Memo AIM-285, Stanford, CA. (August, 1976).

8. Binford, T.O., Liu, C.R., Gini, G., Gini, M., Glaser, I., Ishida, T., Mujtaba, M.S., Nakano, E., Nabavi, H., Panofsky, E., Shimano, B.E., Goldman, A., Scheinman, V.D., Schmelling, D., Gafford, T. "Exploratory study of computer integrated assembly systems", Progress report 4, Stanford Artificial Intelligence Laboratory Memo AIM-285.4, Stanford, CA. (June, 1977).

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

10. Grossman, D.D., Taylor, R.H. "Interactive generation of object models with a manipulator", Stanford Artificial Intelligence Laboratory, Memo AIM-274, Stanford, CA. (December, 1975).

11. Reiser, J.F. (editor) "SAIL", Stanford Artificial Intelligence Laboratory Memo AIM-289, Stanford, CA. (August, 1976).

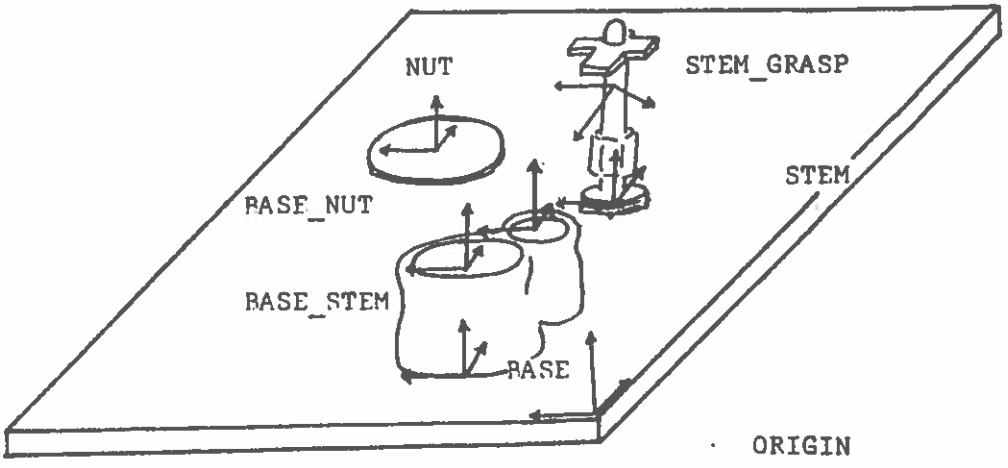

Fig. 3.1 - The assembly station.

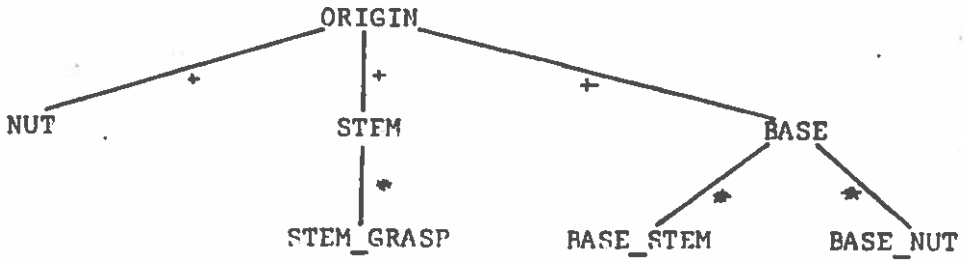

Fig. 3.2 - The frame tree.

```
WORLD (NILROTN,NILVECT)                                    BHAND 1.60
 -ORIGIN (NILROTN,(5.86,25.7,.290))                        YHAND .000
   +NUT (NILROTN,(14.5,10.0,.000))
   +STEM (NILROTN,(10.5,2.00,.000))
      STEM_GRASP ((Y,111.)*(Z,-3.63),(-.457,-.135,1.87))
   +BASE (NILROTN,(2.60,2.00,.000))
      BASE_NUT (NILROTN,(1.60,.000,2.40))
      BASE_STEM (NILROTN,(.150,.000,2.16))
 +YARM (NILROTN,NILVECT)
 +BARM ((Y,12.1)*(Z,2.15),(10.0,27.7,2.69))


 NILTRANS (NILROTN,NILVECT)
```

*0 DECLAR.AL	NILROTN (Z,.000)	NILVECT (.000,.000,.000)

Fig. 4. 1 - The world model on the display.