Languages for Sensor-Based Control in Robotics

Edited by

Ulrich Rembold   Klaus Hörmann

NATO ASI Series

Series F: Computer and Systems Sciences, Vol. 29

Languages for Sensor-Based Control in Robotics

Edited by

Ulrich Rembold
Klaus Hörmann

Institute of Computer Science III
Robotics Research Group
University of Karlsruhe
7500 Karlsruhe/Federal Republic of Germany

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo
Published in cooperation with NATO Scientific Affairs Divison

Proceedings of the NATO Advanced Research Workshop on Languages for Sensor-Based Control in Robotics held in Il Ciocco, Castelvecchio Pascoli/Italy, September 1–5, 1986.

ISBN 3-540-17665-9 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-17665-9 Springer-Verlag New York Berlin Heidelberg

Library of Congress Cataloging in Publication Data. NATO Advanced Research Workshop on Languages for Sensor-Based Control in Robotics (1986 : Castelvecchio Pascoli, Italy) Languages for sensor-based control in robotics. (NATO ASI series. Series F, Computer and systems sciences ; vol. 29) Proceedings of the NATO Advanced Research Workshop on Languages for Sensor-Based Control in Robotics held in Il Ciocco, Castelvecchio, Pascoli/Italy, Sept. 1–5, 1986. 1. Robotics—Congresses. 2. Robots—Programming. 3. Programming languages (Electronic computers)—Congresses. I. Rembold, Ulrich. II. Hörmann, Klaus. III. Title. IV. Series: NATO ASI series. Series F, Computer and systems sciences ; vol. 29.
TJ210.3.N37 1986 629.8'92 87-4950
ISBN 0-387-17665-9 (U.S.)

Printed in Germany

Printing: Druckhaus Beltz, Hemsbach; Bookbinding: J. Schäffer GmbH & Co. KG, Grünstadt
2145/3140-543210

Languages for Sensor-Based Control in Robotics

NATO ASI Series

Advanced Science Institutes Series

A series presenting the results of activities sponsored by the NATO Science Committee, which aims at the dissemination of advanced scientific and technological knowledge, with a view to strengthening links between scientific communities.

The Series is published by an international board of publishers in conjunction with the NATO Scientific Affairs Division

A Life Sciences **B Physics**	Plenum Publishing Corporation London and New York
C Mathematical and Physical Sciences	D. Reidel Publishing Company Dordrecht, Boston, Lancaster and Tokyo
D Behavioural and Social Sciences **E Applied Sciences**	Martinus Nijhoff Publishers Boston, The Hague, Dordrecht and Lancaster
F Computer and Systems Sciences **G Ecological Sciences** **H Cell Biology**	Springer-Verlag Berlin Heidelberg New York London Paris Tokyo

Series F: Computer and Systems Sciences Vol. 29

PREFACE

Robot technology is one of the most promising future tools in manufacturing and will have a significant economical impact on the efficient use of production equipment. Experience has shown that this technology is very difficult and expensive to develop. This problem has been recognized and much research is currently being done in various countries in order to obtain fundamental software and hardware tools for robot planning and control. However, the staggering cost of these developments makes a multinational effort necessary to pool resources. The purpose of this workshop is to define the state of the art of languages and related issues for robots and to give recommendations for future research.

Modern explicit robot programming languages were developed from different technologies. The first robots were programmed by inserting connector pins into a matrix plugboard. This method was followed by teaching the action of a robot by guiding the effector by hand through the desired trajectory. For this purpose the teach pendum was introduced. All these methods were successful for implementing simple applications. However, when robots were used for more complex tasks such as following a defined trajectory or for difficult assembly operations, explicit programming languages had to be developed. Similar to conventional computer programming languages, robot programming languages followed the development stage from the machine language to the assembly language and finally to the compiler language. There are presently over 100 different types of robot programming languages available. Most of them are robot specific. They have limited facilities for handling sensors, different number of axes, gripper motions and concurrent operations. Almost all explicit languages are cumbersome to use and in many cases they require thorough computer science knowledge.

Explicit programming languages can be enhanced by graphical simulation tools. With this method, the motion of the robot and that of the effector are animated on a graphical screen, and the programmer may control and observe the action of the robot.

A user-oriented approach for robot programming is to direct the motions of the robot with task-specific instructions. This method is of particular interest for planning and programming of assembly work. However, assembly tasks are usually so complex that the robot must be equipped with a comprehensive sensor system, for example consisting of vision, approach, touch and force-torque sensors. The interpretation of the implicit robot instructions and the coordination of the robot control and sensors must be done by an expert module which is an essential part of the robot programming system. The user communicates with the implicit programming system by speech or formalized input in a task-oriented mode. A comprehensive expert system will contain the following knowledge:

- how to interpret the task-oriented instructions and how to convert them into an explicit program
- how to describe to the robot its own physical abilities and limitations
- how to describe to the robot the workpiece, its workplace, and tools
- how to plan assembly operations
- how to cooperate with other robots, machine tools and material movement equipment

- how to recognize and solve conflicts
- how to observe robot actions and changes of the workpiece by using different sensors and vision
- how to coordinate all actions of the system and the use of expert knowledge

This book contains the revised proceedings of the NATO International Advanced Research Workshop, which was held in Castelvecchio Pascoli, Italy, from September 1-5th, 1986.

The papers presented in this workshop have proposed methods and tools for the development of task-oriented programming systems, including geometric modeling, simulation, knowledge-based approaches and motion planning. A final session was devoted to mobile robots.

In addition to the lectures, sessions of working groups were organized to discuss selected topics of robotics and to give recommendations for future research work. The reports of the working groups are included in the appendix.

The editors would like to thank the authors of the papers and the participants of the workshop for their excellent contributions.

Prof. Alain Liegeois, Prof. Peter Lockemann, Prof. Marco Somalvico, and Prof. Richard Volz helped to organize the workshop and to select the papers. Dr. Mario di Lullo and Dr. L. V. da Cunha and the members of the Robotics Panel initiated the workshop and gave valuable assistance during the preparation phase. The editors would like to express their gratitude to all people who contributed to the success of the workshop.

Financial support of the workshop was granted by the NATO Scientific Affairs Division. The participants would like to thank the NATO for making this workshop possible.

Karlsruhe, September 1986

Ulrich Rembold
Klaus Hörmann

TABLE OF CONTENTS

MOBILE ROBOTS

APPENDIX

Session

ROBOT PROGRAMMING

Programming of Industrial Robots

Today and in the Future

Prof. Dr.-Ing. Ulrich Rembold
Institute of Computer Science III
Robotics Research Group
University of Karlsruhe
7500 Karlsruhe, F. R. Germany

Abstract

Industrial robots are being used for jobs with ever increasing complexity. Consequently the writing of application programs for robots is becoming very time consuming and expensive. Originally the teach-in method was the principal means of robot programming. The next phase was the development of explicit textual programming systems. However, these languages did not stand the test of being practical since every move has to be described explicitly and for this reason, programming is very involved. Furthermore for applying these languages, good computer science knowledge is necessary. Presently implicit or task-oriented languages are being developed by which the work of the robot is described in an application-oriented fashion. With this tool the user will have a programming method, with which the problem can he described by a natural language. For the development of such systems much basic research is necessary and expert systems are needed which can produce task-oriented solutions with the help of domain knowledge. In this contribution the present state of robot programming languages is described and an overview of the development of future programming methods is given.

1. Historic Development of Programming Languages for Robots

Despite of the fact that programming languages for robots have been development for almost two decades, they have not been generally accepted as it was the case with NC, scientific and commercial languages. Present robot programming language and methods

NATO ASI Series, Vol. F29
Languages for Sensor-Based Control in Robotics
Edited by U. Rembold and K. Hörmann

are tailored either to the application problem, the design of a specific robot, or the type of sensor system used by the robot control system. The historic development of robots, starting with simple pick and place devices and leading to the present sophisticated devices, has been accompanied by the concurrent development of programming methods with increasing power and complexity. The first robots were programmed with plug boards. This method was later improved by the teach-in process. Further improvements were achieved by the incorporation of velocity, time, program branching and many other special functions. The next improvement was the introduction of textual programming.

In textual programming, the operating sequence of the work trajectory of the robot and effector is written with textual instructions. During the initial development phase such languages were obtained by extending existing programming languages with robot specific instructions. The special robot programming languages were developed later.
A program written in such a language, must be translated by a system program which may be either an interpreter or a compiler.
With an interpreter system, a program is decoded, instruction after instruction and is executed stepwise by the robot control. In a compiler system the complete program is translated (compiled) into an intermediate code and is subsequently executed by an interpreter program. Direct interpretable languages do not achieve the power of compiler languages, however, they result in less expensive hardware and software for the robot. Today textual programming languages are mostly based on the interpreter principle. An essential characteristics of textual programming is that the program can be developed off-line, without the use of the robot. The trajectories are described textually and translated by a program development system.
There are two different methods being used :

- With the first method the specification of the coordinates for the trajectories must be defined explicitly in the program in numerical form as constants or variable values. This means that the coordinates must be either measured or read from a design drawing or obtained from a database and than inserted into the program. This method is very complicated and is therefore seldom used.
- With the second method the trajectory is described only through symbolic identifiers. The values of the variables used for the start, intermediate and end points, as well as the effector orientation must not necessarily be known at the time of programming. Concurrently with the programming, the move parameters are taught to the robot with the help of a teach-in system and they are inserted in the textual program during the program execution (**fig.** 1). This method is a pure textual programming procedure. Contrary to the first method, it offers a great advantage in its comfortability and visibility of geometric relationships.

In addition to the motion commands, the robot languages are frequently supplemented with the following features :

- Elements for the description of the geometry of the workpieces and their geometric relations between one another.
- Data types and language elements for geometric calculations.
- Language elements for the interaction with sensors and other peripherals.
- Some real-time attributes and concepts for the realization of parallel processes.

There is a great difference between the power and versatility of the languages which have been developed to date. The scale ranges from the simplest assembler level to modern high level languages.

A classification of some typical languages according to their capability is shown in **fig.** 2 [Bonner 82] . Most of the languages are designed for explicit programming, i.e., each movement must be explicitly described by the programmer. The number of instructions to be programmed for a task is usually very large even for the case of a simple problem. Currently, there exist only a few implicit language concepts, among which one is AUTO-PASS. With this it is possible to specify simple instruction as " pick up the bolt and insert it in the hole ", where every individual move instruction resulting in a move of the robot need not be given.
Fig. 3 and **fig.** 4 show the attributes of selected robot programming languages. The type of the arm configuration, the number of axes and the type of the sensors used are of interest. In some cases the language may be used to program several different arms or arm configurations. At present there exist more than 100 different kinds of robot programming languages. Almost all of them are designed for special robot types and configurations. An advantage of textual programming is that the program is readable by human beings and that it can easily be changed, documented and extended. A second major advantage is the high power of the textual languages. In case of very complicated applications, as for example, with sensor oriented assembly it must be possible to react instantaneously to changing work conditions which usually result in involved program internal calculations and branching.
A disadvantage is the expensive hardware and software required for textual programming. A second disadvantage frequently mentioned is that the textual programming process is hard to follow since the writing and testing of the code is done independently of the robot. Also a highly qualified programmer is needed. This objection can be rebutted since textual programming is mainly used for the solution of complex tasks which can not he solved by other means. With structured programming languages that are based mainly on ALGOL and PASCAL all essential functions of the basic language are retained. In addition extensions are provided which are necessary for robot programming. Some typical additional attributes are :

- Geometric data type as Vector, Rotation and Frame.
- Parallel, cyclic or time-bounded processing of program parts.
- Move instructions with different interpolation and move parameters, e.g. speed, acceleration, etc..
- Parallel instructions for two cooperating robots.
- Specification of different trajectories.

- Instructions for the manipulation of the effectors and the tool system.
- Processing of sensor signals.
- Instructions to control the signal flow of Input/Output interfaces.

By observing **fig.** 2, the reader will be able to specify different programming levels. An attempt is made in **fig.** 5 [Rembold 85] to design a programming system using this layer concept. For example the output of the implicit programming language is explicit SRL code and the output of the SRL program is IRDATA code.

Since most manufacturers of robots use for their devices an especially designed programming language, the presently available languages are differing in various aspects concerning syntax, program structure and features. This fact results in the following disadvantages to the user of different robots:

1. Every robot system used in a facility needs an especially trained programmer.
2. The process of transferring an existing robot program to another system is equivalent to implementing the robot task a second time, even if the target robot has similar kinematics.
3. There is no possibility of transferring taught locations from one robot control to another one.

A solution to this problem could be obtained by a standardized interface between the robot programming system and the robot control system as it is shown in **fig.** 6.
A standard interface many also be moved to a higher level, for example SRL could serve as a common interface (**fig.** 5).

This would provide the following advantages:

1. Programs written on a special programming system could be used to control robots of different types.
2. A robot can use programs or data from any programming system.
3. An expert for programming of a special robot would become a general expert for robot programming.

The resulting flexibility in the use of a robot system and the possibility of reconfiguring a robot cell in an easy way would improve both the efficiency and the performance of a robot installation. Since the interface is concerned with problems of robot programming, the control and the mechanics it must be specified by experts of the different fields. In Germany such a standardization effort is being undertaken. However, a standard must be developed worldwide.

2. Modern Textual Programming Languages

A hierarchical concept of a programming system is represented in **fig. 7**. The upper part represents the implicit programming system which is based on expert systems [Rembold 86] . The lower part represents the basic implementation methodologies for explicit programming, developed in an ESPRIT project. For example the explicit programming language used at the University of Karlsruhe is SRL, and the virtual robot language is IRDATA. Other languages used in the project are LIPAR and the RCM. A common explicit programming language as an interface to the implicit programming system will be introduced at a later project phase, see right side of **fig. 7**.
In addition to the programming languages, a comprehensive simulation system is being developed.

Based on a practical and fundamental study of the VAL and AL systems and the results of a comparative evaluation of many existing languages and concepts of programming systems for industrial robots, the new language SRL was developed [Blume 83] . It has the following features:

1. generalization of the geometric data types, viz., vector, rotation and frame with the help of the structured data types of PASCAL
2. a time compound statement to distinguish between a compound statement for syntactical reasons and a block and sequence of statements which have to be executed sequentially
3. general structure for parallel, cyclic or delayed execution of program parts
4. input-output to digital or analog ports and sensors
5. specification of system components like robots, sensors or interrupts
6. general sensor interface
7. several move statements for different kinds of interpolation

The frame concept was basically taken from AL. Several of the other improvements resulted from the fact that AL was designed hardware dependent. For example, AL can only process sensor data taken from a force and torque measuring box during the execution of a move statement.

A strong effort has been made to overcome hardware dependence and to support structured and self-documented programming. As a new facility, SRL has an interface to a general world model during program run time. The world model may contain data about objects and their attributes like workpieces, fixtures, robots, frames and trajectories.

Another feature of SRL is that it is based on the language PASCAL. The data concept and file management is taken from PASCAL because it gives the user a very flexible and problem oriented data structure.

The goal of the development of SRL was to design a language which can easily be