ON ROBOT PROGRAMMING AND LANGUAGES

A. M. Al-Qasimi, M. Akyurt and F.M. Dehlawi
College of Engineering, King Abdulaziz University, Jeddah

ABSTRACT. A review of developments in the area of robot programming is given. Manual and powered lead-through methods are discussed, along with programming issues like the planning of manipulator trajectories, speed and workcell control, and branching. The main limitations of the lead-through methods are stated.

High level programming languages for industrial robots are introduced. The concepts of robot-oriented programming and task-level programming are discussed. The outstanding features of high-level languages are summarized. New developments to be expected from the next generation of languages are stated.

1. INTRODUCTION

Robot programming is concerned with teaching the robot its work cycle. It involves activities such as defining the motion paths within the workspace, interpreting sensory information, actuating the end effector, sending signals to and recieving data from other devices, and making computations and decisions about the work cycle. There are several methods used for programming a robot; they basically fall into one of the two categories discussed below.

2. LEAD-THROUGH METHODS

In the early 1960's robot programming started by using a teach-by-showing method. It requires the programmer to physically move the manipulator through the desired motion path which is then commited to memory by the robot controller. There are two methods to accomplish this.

2.1 *Manual Lead-through (Walk Through)*

In this method the programmer physically grabs the robot arm and manually moves it through the desired motion/cycle. This is useful for defining a continuous path, where the work cycle involves smooth complex curvilinear movements of the robot arm, such as used in spray painting, or in continuous arc welding. The motion cycle is divided into hundreds or even thousands of individual, closely spaced, points along the path, which are then recorded in memory.

2.2 *Powered Lead-through*

In this method, a teaching pad, which is a hand-held control box with toggle switches, dials and buttons, is used to power-drive and regulate the robot's physical movements and programming capabilities. This is the most commonly used method for programming

today's robots. It is limited, however, to point-to-point motions rather than continuous movement because of the difficulty involved in using the teaching pad to control complex geometric motions in space. The reason this method is so widely used is that many robot applications consist of point-to-point movements, as in movement of parts, and in spot welding.

In both the manual and the powered lead through cases a series of points in space are recorded into the controller's memory for subsequent playback during the work cycle. The controller in such systems has two modes of operation; a *teach* mode for programming, and a *run* mode for executing the program.

2.3 *Programming issues involved*

2.3.1 ***Planning of manipulator trajectories*** Given two endpoints of a path in space, obstacle constraints, and path constraints, the robot controller must come up with an approximation of the desired path consisting of a sequence of time-based control-set points of movement from start to destination endpoints of the path. This can be done by interpolation, using a class of polynomial functions. The path endpoints can be specified either in *Joint-coordinates*, or in *Cartesian coordinates*. It is easier to visualize the correct end-effector configurations in Cartesian coordinates than in joint-coordinates. Besides, joint-coordinates are not orthogonal in most manipulators and do not separate position from orientation. Because of this, Cartesian coordinates are commonly used to specify the endpoints of a manipulator path, and a coversion routine can be used if joint-coordinates are desired.

There are a number of possible trajectories between two endpoints in space, depending on the motion interpolation method specified by the programmer. The possibilities include

- Joint interpolation, where the manipulator is moved along a smooth polynomial trajectory satisfying the position and orientation constraints at both endpoints [1].
- Straight line interpolation, where the manipulator is moved along a straight line path connecting both endpoints [1].
- Circular interpolation, where the programmer is required to specify a circle in the workspace by giving three points on the circle. The controller computes a series of points that can approximate the circle in short straight line segments.
- Irregular motion path is used in manual lead-through to specify a continuous path. Points are used for interpolation to generate thousands of points to be followed.

Methods for computing joint and straight-line interpolations are given in [1]. The algorithms used in the computaions, however, depend on the anatomy of the robot, i.e., polar, cylindrical, Cartesian, or jointed arm [2].

2.3.2 ***Speed control*** The speed at which the robot manipulator will run can be specified during programming by using the dials on the teaching pad to set the speed of different parts of the program. In the lead-through method, the speed is not usually given as a linear velocity at the tip of the end effector because it depends on the number of moving axes at one time, the current axis configuration, and the load carried by the manipulator. This leads to computational complexities in determining the velocity.

2.3.3 ***Workcell control*** The activities of various equipment in the robot workcell must be coordinated to avoid hazards, errors, and the like. A robot controller would have predefined input and output lines of communication carrying signals between the controller and sensors and/or other external devices in the workcell. Push-buttons on the teaching pad may exist for inserting three basic coordination commands.

- SIGNAL M: instructs the controller to output a signal through output line M.
- WAIT N: The robot must wait at current location until a signal on line N is received.
- DELAY X: The robot should wait X seconds before proceeding to the next program step.

2.3.4 ***Branching*** This capability allows the programmer to divide a program into one or more sections, each of which can be thought of as a subroutine that can be called one or more times during program execution. Most controllers allow a branch to be assigned one name from an established group of names, enable incoming signals to invoke a branch, and allow the programmer to specify whether the signal should cause the currently executing branch to be interrupted or wait until it is completed. Executing a branch could then be driven either by a sensor signal, a signal from an external device, or by directly calling the branch by name from within the program. Branching allows more than one task to be programmed into one robot and signals from sensors or other external devices to be used for activating the appropriate branch for the task. Interruption could also be used to activate error handling branches.

Relocatable branching allows the programmer to specify a branch involving a set of incremental points in space relative to some defined starting point for the branch. This allows the same set of movements to be performed at different locations in the work place. Methods for specifying branching and branch definitions vary from one robot to another, but a branching capability would boost the programming efficiency, making it easier to edit, and result in shorter programs.

2.3.5 ***Limitations of Lead-through Methods*** The limitations of these methods are basically threefold. The robot can not be used in production while it is being programmed. It becomes increasingly difficult to program a robot as task complexity is increased. This kind of programming is not compatible with modern computer-based technologies such as CAD/CAM systems, networked data communications, and CM (Computer Manufacturing) systems.

3. HIGH LEVEL PROGRAMMING LANGUAGES

Most of the robot programming languages of today use a combination of textual highlevel programming to define the logic and sequence of the program, and teaching pad programming to define the specific point locations in the workspace. The first high level robot language was developed in 1973 as an experimental language called WAVE at the Stanford artificial intelligence laboratory. It demonstrated the feasibility of robot hand-eye coordination [2].

Development of a subsequent language called AL began at Stanford in 1974. AL could be used to control multiple arms in tasks requiring arm coordination. In 1979, the first commercially available robot high level language called VAL was introduced by Unimation, Inc. It included many of the concepts of WAVE and AL. Later, in 1984, it

was upgraded to VAL II.

In 1976, IBM started developing a high level language for robots. The result was two languages, AUTOPASS and AML. In 1982, AML was made commercially available for IBM robotic products. Other languages include RAIL, introduced in 1981 by Automatix for robot assembly and arc welding; MCL, which was developed by McDonnel-Douglas as an enhancement to APT, the numerical-control part-programming language; and the robot programming language, HELP, available from the General Electric company.

All languages mentioned above fall into one of two main robot programming language categories according to their programming approach. Those categories are:

3.1 *Robot-oriented Programming*

In this category, an assembly task is explicitly described as a sequence of robot motions. Each program statement corresponds, roughly, to one robot action. Such languages are usually built as extensions to an existing ordinary high level programming language, such that robot programming requirements are met. Most current robot programming languages are of this type. They are characterized by the steps taken to develop a robot program:

A. The workspace is setup, and parts are fixed by use of fixtures and feeders.
B. Location of parts, specified as an orientation and a position, and part's features are defined, using the data structures provided by the language.
C. The required task is partitioned into a sequence of actions, like move, grasp, etc..
D. Sensory commands are added to detect error conditions and monitor the progress of a task.

As tasks became more and more complex, the programming process became increasingly difficult because too many details are required to be provided by the programmer.

3.2 *Task-level Programming*

In this category, an assembly task is described as a sequence of positional goals of objects rather than the motions needed to reach these goals. So there is no explicit robot motion to be specified. The programming approach is simpler than before. The steps involved are also the characteristics of these languages:

A. *World Modeling*, which is to form a data base describing the geometric and physical properties of the objects, including the robot itself, and their assembly state in the workplace.
B. *Task Specification*, as a sequence of states of the world model, or as a sequence of symbolic operations on the objects.
C. *Robot Program Synthesis* is computationally similar to the idea of automatic program generation. Given a world model, and a task specification, an automatically generated robot program is required.

The first and second steps are to be done by the programmer, while the third step is to be performed by the programming language processor. It is one of the most difficult phases of producing a robot program. Research in this area is still active to solve many of the difficulties involved, such as task planning, grasp planning, trajectory planning, obstacle avoidance, and sensory information utilization. Languages of this type are few,

and most of them lack something in the synthesis part. Task level programming languages include AUTOPASS and AL.

3.3 *Features of High-level Languages*

Regardless of the programming approach taken, a robot programming language of today must have built-in modules or commands offering the following features:

- Motion control, to define manipulator motions, speed, interpolation method(s), branching, and sensor commands.
- Sensor capabilities, to deal with binary and analog signals, and control external devices using those signals.
- Intelligence, to be able to modify system behaviour in a programmed manner depending on information about work environment, such as error recovery and the like.
- Communication and data processing, to interface and communicate with other computers and data bases for the purpose of keeping records, generating reports, and controlling workcell activities.

Additional features that may be incorporated in a robot programming language include real-time processing, parallel, distributed, and pipelined architectures and algorithms, artificial intelligence, advanced sensory information, such as machine vision and human voice recognition, neural networks for adaptive learning, and graphical user interfaces with graphical and virtual reality programming and simulation.

3.4 *Next Generation Languages*

The next generation of robot programming languages would be of the task-level type, where the world model information is entered automatically using vision systems in the workcell, and program synthesis is done entirely by the computer. The programmer would only specify the task to be done either by voice or by written command of a high level, such as "Assemble Computer". Advanced techniques such as artificial intelligence, neural networks, and object oriented programming/database systems and others may be used.

4. LITRATURE ON ROBOT PROGRAMMING

Robot programming is a very active research subject, published litrature span a wide range of related topics. Here is a sample of recently published work. On the subject of robot-level programming more information is to be found in [1-28], while task-level programming is discussed in [1, 29-41]. A brief description and a list of comparative features of some current robot programming languages are presented in [42]. Recent progress made in robot programming and task planning systems until 1990 is reviewed in [43]. Off-line robot programming is an added advantage to a language because the robot may be reprogrammed without having to be stopped for a long time. Some of the languages used for off-line programming are given in [23, 25, 44-46].

Trajectory planning using neural networks is investigated in [47]. A parallel processing approach for finding the shortest path between two endpoints using the wave-front technique is reported in [48, 49]. A real-time trajectory generation technique for the Multi-RCCL robot programming system is to be found in [50]. Automatic path planning for several arms handling one object and avoiding obstacles by moving the object from

one arm to another is given in [51]. Fuzzy expert systems and evolutionary algorithms, integrated to produce Fuzzy Evolutionary Algorithms, FEA, for automatic trajectory generation are presented in [52].

Languages for describing objects are discussed in [53, 54]. A neural network model for acquiring data from a solid modeling data base coupled with the uncertainty of the grouping process that was used to perform the geometric classification of objects is presented in [55]. A generic software safety verification and encoding language for safety critical robot actions are given in [56].

Presentations of newly developed robot programming environments are in made in [9-10, 25, 37-38, 45, 50, 57-59]. An offline programming system for programming, analysis and interactive computer simulation of a general five axes manipulator is described in [44]. Obstacle avoidance algorithms embedded in programming languages can be found in [60-63]. The extending of programming languages for multi-robot coordination and programming is presented in [64]. An off-line robot programming system, called ROPSIM, acting as a true CIME-subsystem and allowing the reuse and exchange of robot model definition data and program definition data with systems of other origins and different functionalities is presented in [45].

In task planning, a configuration space approach for moving an object through a crowded work space is presented in [65]. An elastic network approach for task planning and obstacle avoidance is presented in [66]. An optimization technique for minimizing a criterion function derived from the difficulty of state transitions is used for assembly planning in [67]. An artificial intellegance approach for the automatic assembly planning is to be found in [68]. A review of recent research in task planning can be found in [43].

Agent-based programming in which small software units are interacting in parallel is found in [59], and a language based on that, called SAL (the SmartyCat Agent Language), is presented in [69]. Some issues concerning agent based systems and parallel programming are discussed in [70].

Graphical environments for robot programming make it easier to program and simulate a robot. In [11], a behaviour-based graphical robot simulator, JC-3, is used to test the "meta software" generated by an interactive graphical "agent editor". An extension to the C language, called SMALL, gives a graphical environment for robot programming [37], and virtual reality is used to teach a robot an assembly task in virtual space [36].

The object-oriented paradigm is used in [16-17, 38], and a concurrent object-oriented real-time robot programming language, called RSPL (Robot Schema Programming Language), is described in [19].

Artificial Intellegence techniques are widely used for task-level and mobile robot behaviour programming. For example, robot programming by human demonstration is discussed in [32, 33]; Sensor based robot programming by creating robot skills is dicussed in [71]; Redundant robots trajectory planning and programming based on relaxation networks, M-Nets, and the NEM++ language is presented in [72]; And a method for behaviour specification in an unstructured environment for a simple robot insect is reported in [73].

Neural networks are also being increasingly used to control and train robots, a sample of the recent litrature describing their use is given in [74-92]. A neural network artificial brain for controlling around 1000 behaviours in a "robot kitten" is reported as part of the "CAM-Brain" project in [74]. A study showing the potential of neural networks in mobile robot applications is given in [76]. A software tool, called ANNECS, for compiling a high level object-orientd specification into a functionally equivalent neural network is described in [82]. Methods for learning the robot's inverse kinematics using neural networks can be found in [83, 92].

5. CONCLUDING REMARKS

It is clear from the discussion above that programming of industrial robots and the related languages have witnessed fundamental developments in the past two decades. It is possible to predict that the developments to take place during the coming two decades will enable the industrial robot to become a natural component of the manufacturing environment. The challenges that remain are to render the programming of robots as a *user friendly* exercise, and to bring down the initial cost of the robotic system, without, naturally, sacrificing from the power of flexibility that today's industrial robots possess.

REFERENCES

1. Fu, K.S., Gonzalez, R.C., and Lee, C.S.G., Robotics: control, sensing, vision, and intelligence, McGraw-Hill, 1987.
2. Groover, M.P., Weiss, M., Nagel, R.N., and Odrey, N.G., Industrial Robotics: technology, programming, and applications, McGraw-Hill, 1986.
3. Bonner, S. and Shin, K.G., "A comparative study of robot languages," IEEE computer, vol.15, no.12, 1982, pp. 82-96.
4. Gruver, W.A., et al., "Industrial robot programming languages: A comparative evaluation," IEEE Trans. Systems,Man,Cybern. vol.SMC-14, no.4, 1984, pp. 321-333.
5. Paul, R.P., Robot Manipulator: mathematics, programming, and control, MIT Press, 1981.
6. Snyder, W.E., Industrial Robots: computer interfacing and control, Prentice-Hall, 1985.
7. Hong, T., Xia, K., Xu, W., Chen, C., Lu, L., "Six-joint industrial robot controller software system," Proc. TENCON '93, IEEE Region 10 Conference on 'Comp., Comm., Control and Power Engg.', vol.4, 1993, pp. 182-185.
8. Lees, D.S., Leifer, L.J., "A graphical programming language for robots operating in lightly structured environments," Proc. IEEE Intl. Conf. on Robotics and Automation, vol.1, 1993, pp. 648-653.
9. Matsumoto, A., Ando, M., "Interactive robot programming system for educational use," Proc. IEEE Intl. Workshop on Robot and Human Communication, 1992 pp. 419-424.
10. Rees, J., Donald, B., "Program mobile robots in Scheme," Proc. IEEE Intl. Conf. Robotics And Automation, vol.3, 1992, pp. 2681-2688.
11. Ojala, J., Inoue, K., Sasaki, K., Takano, M., "Interactive graphical mobile robot programming," Proc. IEEE/RSJ Intl. Workshop Intelligent Robots and Systems, vol.3, 1991, pp. 1485-1490.
12. Duhaut, D., Monacelli, E., "Including control in the definition of a programming language for multi-robots," Proc. IEEE/RSJ Intl. Workshop Intelligent Robots and Systems, vol.3, 1991, pp. 1382-1387.
13. Duhaut, D., Bidaud, P., Fontaine, D., "IAda. A language for robot programming based on Ada," Robotics and Autonomous Systems, vol.9, no.4, 1992, pp. 299-304.
14. Liang, L., Crangle, C., Leifer, L., "A computational model for a robotic arm instructed by natural language," Proc. IEEE Intl. Conf. Systems, Man and Cybernetics, 1990, pp. 451-456.
15. Gat, E., "ALFA: a language for programming reactive robotic control systems," Proc. IEEE Intl. Conf. Robotics and Automation, vol.2, 1991, pp. 1116-1121.
16. Boyer, M., Daneshmend, L.K., Hayward, V., Foisy, A., "An object-oriented paradigm for the design and implementation of robot planning and programming systems," Proc. IEEE Intl. Conf. Robotics and Automation, vol.1, 1991, pp. 204-209.

17. Hayward, V., Daneshmend, L.K., Foisy, A., Boyer, M., Demers, L.P., Ravindran, R., Ng, T., "The evolutionary design of MCPL, the MSS command and programming language," Proc. IEEE Intl. Workshop on Intelligent Robots and Systems, vol.1, 1990, pp. 413-420.
18. Blomme, R.M., Van Campenhout, J.M., "Asynchronous parallel programming techniques for compliant robot motions," Second Intl. Conf. on Software Engineering for Real Time Systems, 1989, pp. 204-208.
19. Pocock, G., "A distributed, real-time programming language for robotics," Proc. IEEE Intl. Conf. on Robotics and Automation, vol.2, 1989, pp. 1010-1015.
20. Elmaghraby, A.S., "A robot control language," Conf. Proc. IEEE SOUTHEASTCON, 1988, pp. 413-416.
21. Moore, G., "Robot programming: the language of labour?," Electronics and Power, v.31, Jul. 1985, pp. 499-502.
22. Rony, P.R., Rony, K., "Introduction to robot programming in BASIC (book review)," The Industrial Robot, v.12, Dec. 1985, p. 271.
23. Woodcock, R., "Robot Basic integrates functions to facilitate off-line programming," Electronics, vol.57, July 12, 1984, pp. 124-127.
24. Holmes, D.S., "EARL; an easy robot language," Robotics Age, vol.6, Nov., 1984, pp. 20-21.
25. Fujiuchi, M., Nakamura, T., Yamaguchi, M., Ishiguro, Y., Mizutani, S., Nakano, M., "Development of a robot simulation and off-line programming system," SAE Technical Paper Series, Warrendale, PA, USA, 1992, pp. 69-77.
26. Brown, P.J., An abstract device approach to the programming and monitoring of flexible assembly cells (robotic cells), PhD Dissertation, Council for National Academic Awards, UK, 1991.
27. Bal, B.S., Studies in robot programming (Forth language, Automatic object location), PhD Dissertation, Aston University, UK, 1990.
28. Pauli, D., Adaptive robot training: Explorations in sensorless manipulation, MSc. Dissertation, University of Calgary, Canada, 1990.
29. Binford, T.O., "The AL language for intelligent robots," in Proc. IRIA Sem. Languages and Methods of Programming Industrial Robots, 1979, pp. 73-87.
30. Lieberman, L.I., and Wesley, M.A., "AUTOPASS: An automatic programming system for computer controlled mechanical assembly," IBM J. Res. Devel. vol.21, no.4, 1977, pp. 321-333.
31. Mujtaba, M.S., Goldman, R.A., and Binford, T., "The AL robot programming language," Comput. Engr., vol.2, 1982, pp. 77-86.
32. Kuniyoshi, Y., Inaba, M., Inoue, H., "Learning by watching: extracting reusable task knowledge from visual observation of human performance," IEEE Trans. Robotics and Automation, Vol.10, N0.6, Dec. 1994, pp. 799-822.
33. Delson, N., West, H., "Robot programming by human demonstration: the use of human variation in identifying obstacle free trajectories," Proc. IEEE Intl. Conf. Robotics and Automation, vol.1, 1994, pp. 564-571.
34. Shepherd, B., "Applying visual programming to robotics," Proc. IEEE Intl. Conf. on Robotics and Automation, vol.2, 1993, pp. 707-712.
35. Coste-Mainere, E., Espiau, B., Rutten, E., "A task-level robot programming language and its reactive execution," Proc. IEEE Intl. Conf. Robotics And Automation, vol.3, 1992, pp. 2751-2756.
36. Takahashi, T., Ogata, H., "Robotic assembly operation based on task-level teaching in virtual reality," Proc. IEEE Intl. Conf. Robotics And Automation, vol.2, 1992, pp. 1083-1088.
37. Smith, M.G., "An environment for more easily programming a robot," Proc. IEEE Intl. Conf. Robotics And Automation, vol.1, 1992, pp. 10-16.
38. Miller, D.J., Lennox, R.C., "An object-oriented environment for robot system architectures," IEEE Control Systems Magazine, Vol.11, No.2, Feb. 1991, pp. 14-23.
39. Holton, D.R.W., McKeever, J.D.M., McKeag, R.M., "Formal description techniques in robot programming," IEE Colloquium on Application of CASE Tools, Digest No.058, 1990, pp. 1/1-6.
40. Mahadevan, Sridhar, Connell, J., "Automatic programming of behavior-based robots using reinforcement learning," Artificial Intelligence, v.55, June 1992, pp. 311-365.
41. Koza, J.R., Rice, J.P., "Automatic programming of robots using genetic programming," Proc. 10th Natl. Conf. on Artificial Intelligence, 1992, pp. 194-201.
42. Klafter, R.D., Chmielewski, T.A., and Negin, M., Robotic Engineering: an integrated approach, Prentice-Hall, 1989.
43. Rondeau, J.M., and ELMaraghy, H.A., "Robot programming and task planning," Manuf. Rev., vol.3, no.4, 1990, pp. 245-251.

44. Lee, D.M.A., and ELMaraghy, W.H., "ROBOSIM. A CAD-based off-line programming and analysis system for robotic manipulators," Comput. Aided Eng. J. vol.7, no.5, 1990, pp. 141-148.
45. Nielsen, L.F., Trostmann, S., Trostmann, E., and Conrad, F., "Robot off-line programming and simulation as a true CIME-subsystem," Proc. IEEE Intl. Conf. Rob. Autom., 1992, pp. 1089-1094.
46. Carter, S., "Off-line robot programming: the state-of-the-art," The Industrial Robot, v.14, Dec. 1987, pp. 213-215.
47. Kyung, K., Ko, M., and Lee, B., "A hierarchical neural network structure for robot trajectory planning," Proc. 29th SICE annual Conf., 1990, pp. 833-836.
48. Suzuki, H., and Arimoto, S., "Parallel-processable recursive and heuristic method for path planning," Proc. Intl. Conf. Soil Mech. Found. Eng., 1989, pp. 616-618.
49. Ranganathan, N., Parthasarathy, B., Hughes, K., "A parallel algorithm and architecture for robot path planning," Proc. Eighth Intl. Parallel Processing Symposium, 1994, pp. 275-279.
50. Lloyd, J., and Hayward, V., "Real-time trajectory generation in Multi-RCCL," J. Rob. Syst., vol.10, no.3, 1993, pp. 369-390.
51. Koga, Y., Latombe, J.-C., "On multi-arm manipulation planning," Proc. IEEE Intl. Conf. Robotics and Automation, vol.2, 1994, pp. 945-952.
52. Xu, H.Y., Vukovich, G., "Fuzzy evolutionary algorithms and automatic robot trajectory generation," Proc. First IEEE Conference on Evolutionary Computation; IEEE World Congress on Computational Intelligence, vol.2, 1994, pp. 595-600.
53. Grossman, D.D., and Taylor, R.H., "Interactive generation of object models with a manipulator," IEEE Trans. System,Man,Cybern. vol.SMC-8, no.9, 1978, pp. 667-679.
54. Wesley, M.A., et al., "A geometric modeling system for automated mechanical assembly," IBM J. Res. Devel., vol.24, no.1, 1980, pp. 64-74.
55. Ali, A.L., Ali, D.L., and Ali, K.S., "Undeterministic manipulation of solid models for robot program synthesis," Proc. Conf. Comput. Ind. Eng., vol.19, 1990, pp. 465-468.
56. Rahimi, M., and Xiadong, X., "Framework for software safety verification of industrial robot operations," Comput. Ind. Eng., vol.20, no.2, 1991, pp. 279-287.
57. Stewart, D.B., Schmitz, D.E., and Khosla, P.K., "Implementing real-time robotic systems using CHIMERA II," Proc. Intl. Conf. systems engineering, 1990, pp. 252-257.
58. ELMaraghy, H.A., and Laperriere, L., "Modelling and sequence generation for robotized mechanical assembly," Rob. Autom. Syst., vol.9, no.3, 1992, pp. 134-147.
59. Zanichelli, F., Caselli, S., Natali, A., Omicini, A., "A multi-agent framework and programming environment for autonomous robotics," Proc. IEEE Intl. Conf. Robotics and Automation, vol.4, 1994, pp. 3501-3507.
60. Brooks, R.A., "Planning collision free motion for pick-and-place operations," Intl. J. Robotics Res., vol.2, no.4, 1983, pp. 19-44.
61. Lewis, R.A., and Bejczy, A.K., "Planning considerations for a Roving robot with arm," Proc. 3ed Intl. Jt. Conf. Artificial Intelligence, 1973.
62. Lozano-Perez, T., "Robot programming," Proc. IEEE, vol.71, no.7, 1983, pp. 821-841.
63. Lozano-Perez, T., and Wesley, M.A., "An algorithm for planning collision free paths among polyedral obstacles," Comm. ACM, vol.22, no.10, 1979, pp. 560-570.
64. Tsai, C., "Multiple robot coordination and programming," Proc. IEEE Intl. Conf. Rob. Autom., 1991, pp. 978-985.
65. Lozano-Perez, T., "Task planning," in Robot Motion: planning and control, (M. Brady, et al., eds.), MIT Press, 1983.
66. Wong, W.S., and Funka-Lea, C.A., "An elastic net solution to obstacle avoidance tour planning," Intl. Jt. Conf. Neural Networks, 1990, pp. 799-804.
67. Yoshikawa, T., Yokokohji, Y., and Yu, Y., "Assembly planning operation strategies based on the degree of constraint," Proc. Intl. Conf. Soil Mech. Found. Eng., 1989, pp. 682-687.
68. Garrod, W., and Everett, L.J., "A.S.A.P. Automated sequential assembly planner," Proc. ASME Intl. Comput. Eng. Conf. Expo., 1990, pp. 139-144.
69. Lim, W., "An agent-based approach for programming mobile robots," Proc. IEEE Intl. Conf. Robotics and Automation, vol.4, 1994, pp. 3584-3589.
70. Bozinovski, S., "Parallel programming for mobile robot control: agent-based approach," Proc. 14th Intl. Conf. Distributed Computing Systems, 1994, pp. 202-208.
71. Archibald, C., Krieger, M., Petriu, E., "Software design of sensor-based robot skills," Conf. Proc. 10th Anniversary. IMTC/94, Advanced Technologies in I & M.; IEEE Instrumentation and Measurement Technolgy Conference, vol.1, 1994, pp. 175-178.

72. Franchi, P., Morasso, P., Vercelli, G., Zaccaria, R., "Integrating force-fields methods in a robot planning/programming language," Fifth Intl. Conf. Advanced Robotics: Robots in Unstructured Environments, vol.2, 1991, pp. 1170-175.
73. Mitchell, R.J., Keating, D.A., Kambhampati, C., "Learning strategy for a simple robot insect," Intl. Conf. Control, vol.1, 1994, pp. 492-497.
74. de Garis, H., "CAM-Brain: the genetic programming of an artificial brain which grows/evolves at electronic speeds in a cellular automata machine," Proc. First IEEE Conf. on Evolutionary Computation, vol.1, 1994, pp. 337-339, 339a-b.
75. Salichs, M.A., Puente, E.A., Gachet, D., Pimentel, J.R., "Learning behavioral control by reinforcement for an autonomous mobile robot," Proc. Intl. Conf. on Industrial Electronics, Control, and Instrumentation, vol.3, 1993, pp. 1436-1441.
76. Mohamed, A.S., "Future neuro mobile robots," Second IEEE Conference on Control Applications, vol.2, 1993, pp. 637-642.
77. Thrun, S.B., "Exploration and model building in mobile robot domains," IEEE Intl. Conf. on Neural Networks, vol.1, 1993, pp. 175-180.
78. Fogarty, T.C., "Classifier systems for control," IEE Colloquium on Genetic Algorithms for Control Systems Engineering, Digest No. 1993/130, 1993, pp. 8/1-3.
79. Werbos, P.J., "Neurocontrol and elastic fuzzy logic: capabilities, concepts, and applications," IEEE Transactions on Industrial Electronics, Vol.40, No.2, Apr. 1993, pp. 170-180.
80. Holland, O., Snaith, M., "Q-learning with generalisation: an architecture for real-world reinforcement learning in a mobile robot," Intl. Joint Conf. on Neural Networks, vol.1, 1992, pp. 287-292.
81. Lewis, M.A., Fagg, A.H., Solidum, A., "Genetic programming approach to the construction of a neural network for control of a walking robot," Proc. IEEE Intl. Conf. on Robotics And Automation, vol.3, 1992, pp. 2618-2623.
82. Vellacott, O.R., "ANNECS: a neural network compiler and simulator," Intl. Joint Conf. on Neural Networks, vol.2, 1991, p. 991.
83. Brause, R., "Optimal information distribution and performance in neighbourhood-conserving maps for robot control," Proc. 2nd Intl. IEEE Conf. on Tools for Artificial Intelligence, 1990, pp. 451-456.
84. Jansen, M., Eckmiller, R., "Globally stable neural robot control capable of payload adaptation," Proc. Intl. Joint Conf. on Neural Networks, Part.1, 1993, pp. 639-642.
85. Bachelder, I.A., Waxman, A.M., "Mobile robot visual mapping and localization: a view-based neurocomputational architecture that emulates hippocampal place learning," Neural Networks, vol.7, no.6-7, 1994, pp. 1083-1099.
86. Baloch, A.A., Waxman, A.M., "Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN," Neural Networks, vol.4, no.3, 1991, pp. 271-302.
87. Glasius, R., Komoda, A., Gielen, S.C.A.M., "Neural network dynamics for path planning and obstacle avoidance," Neural Networks, vol.8, no.1, 1995, pp. 125-133.
88. Luebbers, P.G., Pandya, A.S., "Vision-based path following by using a neural network guidance system," Journal of Robotic Systems, vol.11, no.1, 1994, pp. 57-66.
89. Sanger, T.D., "Neural network learning control of robot manipulators using gradually increasing task difficulty," IEEE Trans. on Robotics and Automation, vol.10, June 1994, pp. 323-333.
90. Walter, J.A., Schulten, K.J., "Implementation of self-organizing neural networks for visuo-motor control of an industrial robot," IEEE Trans. on Neural Networks, vol.4, Jan. 1993, pp. 86-95.
91. Cousein, A., "Neural networks for robot control," Third Intl. Conf. Software Engg. for Real Time Systems, 1991, pp. 119-124.
92. Golnazarian, W., Shell, R., Hall, E.L., "Robot control using neural networks with adaptive learning steps," Proc. SPIE - The Intl. Society for Optical Engg., vol 1826, 1993, pp. 122-129.