Communication is problem in programmed control

Part four of Mark Witkowski’s series concerns itself with the programmed control of industrial-style robots and problems of the communication of ideas from the user to the machine.

However brilliantly designed and constructed, and whatever sensory provision is made on a robot device, its usefulness, usability and performance is ultimately dependent on the control algorithms used. Not that one would want to spend a year writing code - assume that robots are all computer-controlled which is, of course, untrue - to compensate for an unstable mechanism, or trying to guess that the world is doing because the robot is insufficiently instrumented. Generally a good robot is made usable by virtue of its programming, a lesser one may be saved.

Deterministic approach

In some cases, the software is so central to an idea that the robot is not built at all, just simulated on a computer graphics terminal. Programming of robots tends to fall into two categories, the deterministic approach, in which the robot is programmed with specific actions known to perform a given task. New ways are always sought to program robots with the minimum of effort for the maximum effect and some have met with success.

When direct teach mode instruction becomes too cumbersome, programming languages are developed to describe the problem and its solution. As there is an incredible mass of detail in the most trivial of everyday tasks, these languages are being developed continually and re-structured to cope. This first category is the province of the industrial robot and industrially-orientated robot research.

The second software category falls within the bounds of artificial intelligence research, where the emphasis is on robot problem solving, and where instead of being instructed in minute detail, only a broad outline or the final goal need be specified, possibly in a natural language. Artificial intelligence techniques are being incorporated slowly into robot programming languages as the tasks the robots are required to perform become more complex and the current intuitive methods are found to be inadequate.

Furthermore, unfortunately, each robot and manipulator tends to have its own teach technique or programming language. There has been little conformity and standardisation, with no universally-accepted language. Unlike different types of computer which, even though they have distinct order-codes, have standardised user languages, manipulators are still sufficiently diverse in design to defeat the compiler writer.

However, numerically-controlled machine tools have been programmed in APT for years - ITT research institute, 1967 - different numerically-controlled machines are catered for by post-processing a universally applicable intermediate codeform from a single compiler into the specific control signals required.

Simplest method

By far the most straightforward method of programming an industrial robot is to manhandle it through the desired sequence of actions. Continuous path robots, as used in paint spraying or welding, are effectively programmed by a skilled human operator leading the arm’s spray/welding head through a complete spray or welding job with the actuator power turned-off. The joint position sensor values are recorded at frequent intervals, either in computer memory, or on tape. When the job is completed to the operator’s satisfaction, the power can be re-applied and the robot will repeat the operator’s actions exactly when the stored data is re-played.

Assuming that one has taken the precaution of placing a new workpiece in precisely the position and orientation of the original, the robot will do as good a job as the man did.

With pick-and-place type robots, the arm is moved to a series of significant positions in the sequence of actions with a joystick control. Although there are many possible designs for this type of control
unit, in principle there will be a switch for each of the degrees of freedom, and a teach button. In a typical application, the arm may be required to move to a component feeder, grasp an item, move to a press and deposit it in place, move out of the way, pick-up the piece after stamping, deposit it in an outgoing hopper and finally return to the feeder to collect the next blank.

This process will involve many discrete steps and the manipulator is moved to each using the multi-switch control. When it is aligned perfectly at each point in the cycle the teach button is pressed, and the joint positions recorded. This is repeated for each significant point, and there may be many before the cycle is re-played to check the sequence.

Specialisations
As the main control unit will compute a straight-line trajectory between the points during playback, it is essential for the user to define sufficient intermediate points to avoid obstacles — none of the actions made between teach points is stored.

There are a number of possible specialisations to this mode of robot instruction by teaching. Figure 1a shows a Visual Programming Device (VPD) used to program the University of Rhode Island (URI) five-degree-of-freedom arm which is shown in figure 1b — Birk and Kelly, 1976, and Kelly and Silvestro, 1977. A computer-compatible TV camera views the base upon which the objects the robot will manipulate are placed. When the VPD is placed on the base, it is possible to calculate the co-ordinates and orientation of the two lights, L1 and L2, from the television image. The VPD is placed round the object to be grasped and the 'P' button pressed on the keypad — figure 1c. During the playback phase, this will have the combined effect of moving to, orientating with, grasping and lifting the selected object. 'R' has the effect moving to, lowering and releasing the object at that specified VPD position. Other commands include 'B', Begin, and define the 'home' position. 'T', move Through a point specified by the VPD, 'E', End, and move back to the home position. 'Wnn', Wait for NN.N seconds, allowing operator intervention. Recorded or memorised points are available, TMn, RMn and PMn for Through, Release at the Pick up at the co-ordinates stored in Memory location n.

This is a particularly useful feature as it is difficult to re-position the VPD repeatedly by hand.

Well-suited
This form of programming is well-suited to the overhead gantry, five-degree-of-freedom manipulator used. A more general six-degree-of-freedom arm would need programming in three dimensions. Perhaps this could be done with stereo television cameras or a navigation-style position sensor.

The visual instruction scheme (V/I) is used in four stages. First the calibration phase. Two fiducial lights, to the left on the base-plate in figure 1b, are used to calibrate the camera co-ordinate-generating program. Recording the sequence of actions using the VPD and keypad is the next stage. Then the Edit/Verify mode is entered. With the aid of a single-step facility, all the points can be checked and changed if they are incorrect.

Misalignment can be corrected by altering one or more of the individual co-ordinate components. Points may be added or, for instance, insufficient clearance was allowed around some obstacle or when some new sub-sequence has been added. Points may be deleted if a path-length proves to be excessively long. When all is as it should be, the arm is put into playback mode and used.

When a manipulator is used as a disabled persons' aid, particular care has to be taken with the design and lay-out of the input mechanism. Todd (1979) describes a multi-mode input cluster with which tetraplegics may operate a manipulator in a number of different ways by head movements alone. He used a ring of 12 photocells suspended in front of a video monitor, which would operate the manipulator when a light beam projector attached to a pair of glasses frame shone on to one of the cells.

Tree structure
Information displayed on the monitor close to each of the photocells labelled their function. Certain cells would have the effect of changing the labels, and hence the effects the Z-80-based controlling microprocessor had on the manipulator. These changes were organised into a tree structure of different modes, including direct control, pre-programmed automatic picking-up and changing to a different input device.

In addition to the photocells, there was also a ring of 12 momentary push switches, arranged in the same manner as the photocells — so that the monitor labels were still useful — operated by a stick held in the mouth. There was also a joystick, operated by placing the lever in

(continued on next page)
removed action node, figure 2b.

Adding a node to the action list is a similar matter of altering the current instruction's pointer to the top of the free list, which is contained in FREE, altering the FREE pointer to the next free node and pointing this new successor node to the old successor node as in figure 2c. Linked list storage allocation is a standard computing technique, about which more can be found in the majority of books on data structures — Knuth, 1968.

A doubly-linked list, in which a second set of pointers point from the successor to current and current to predecessors would allow the actions to be re-played or searched in reverse order. There are doubtsome instances where this would be helpful.

A majority of robots will be programmed by teaching them. It offers a number of important advantages over other methods. There is no need for the operator, who is presumably already skilled in the work the robot is to perform, to understand the intricate detail of robot operation. There is also no need for the operator to learn a specialised programming language, and the machine is ready for use as soon as it is commissioned. Program development and debugging are, therefore, accomplished in the minimum time. Furthermore, there is a minimum of sophisticated equipment in the work-area, at most keypad or joystick control, improving the potential reliability of the whole system.

There are also many disadvantages and limitations to this form of programming, while the robot's action may be performed ad infinitum with no variation all is well, however there are many situations in which a robot should be programmed by telling rather than showing it — Hohn, 1979, Holt, 1979.

Consider the task of picking eggs from a feeder, i.e., a fixed location, passing through some inspection processes, and finally transferring them to a carton — figure 3 — or that of picking the next item of a neatly-stacked pile, each of which is to be found at a position lower than the last.

One fairly bad solution would be to train the whole sequence explicitly. By training the fixed sequence, supply, P1, P2, P3, P4, branch, it could be saved as a macro. Then it would only be necessary to train each of the different branch paths and, after each, press the macro-expand control which substitutes automatically the stored path into the linked-list sequence action queue.

**Real Power**

The real power a programming language gives a robot user is in relation to acting on sensory data. We didn't decide on how to describe the tests on those eggs, or how to dispose of bad ones. As soon as anything more than a few binary interlocks are considered the possible combinations of sensor tests explodes and finding some orderly way of handling the ensuing branch points, feedback loops and error recovery becomes essential.

Subroutine call or macro-expansion can, as with all computing, reduce programming effort considerably, as well as impart a much more modular, top down, control structure to the task solution, particularly where small modifications are required to a basic action sequence. Language makes the description of transformations to the already-programmed actions more powerful. It becomes possible to describe actions relative to some object or previous action, rather than absolute position, or to superimpose the motion of a conveyor belt on which the work-piece is moving.

In some circumstances, it may be desirable to rotate, expand or contract the sequence, or reflect it to give a mirror image. Consider the left- and right-hand sides of car assembly. Where absolute positioning is required, manual control of the action sequence may be insufficiently accurate or repeatable. Print statements in the language are used to provide a written log of robot activity, display messages and sound alarms when operator assistance is needed.
called for. There are many instances where the design criteria for a good robot control language are similar to those of any other type of computer language. They must allow the user to specify every aspect of the task to be performed, without being too cumbersome. Robot languages for manipulators may either describe the task in terms of robot motions or the position and transformations of the work pieces.

WAVE from the Stanford artificial intelligence laboratories — Paul, 1977 — is an example of an industry-oriented manipulator control language. It is written as a sequence of one-line instructions, and is worth closer examination. In WAVE, an object is described by the position the manipulator must be in to grasp it. There are six items required to specify the position and they are assigned to a variable name thus:

TRANS variablename 30,20,10,0,0,0

assigns a particular position (X = 30, Y = 20, Z = 10) in co-ordinate space to the gripper with a unique orientation. The Scheinman arm at Stanford has six degrees of freedom and the latter three parameters to TRANS specify the angle of attack of the gripper completely in relation to fixed reference orientations.

Instructions

MOVE variablename would then cause the manipulator to move from its current position and, assume the co-ordinates and orientation specified in a previous TRANS instruction, which in itself caused no action.

MOVE is an absolute instruction, motions relative to the current position can be made with:

CHANGE vector1,scalar,vector2,angle,time which moves the arm a distance specified by scalar in the direction given in vector 1, also rotating it by angle about vector 2, at an optional speed.

VECT variablename x,y,z is used to specify a vector with x,y and z components, and can equally be used to give a direction or a force heading and value. The gripper is opened and closed with:

OPEN 5
open the gripper to five inches and:

CLOSE 1

close the gripper, the jaws will close until either physical resistance or a specified force is met by appropriate sensors. If they close more than the parameter allows, less than one inch, a well-defined error condition is generated, usually meaning that the object to be grasped was not in the expected position.

CENTRE 1

centres the hand about an object using touch sensors on the insides of the fingers, without moving it. CLOSE and CENTER use sensory data inherently, whereas MOVE and CHANGE do not.

The STOP instruction may be used to abort a movement when a certain, expected, pre-condition is met. So the code:

VECT DOWN 0,0,-1.

next module, or on giving an error message. The 'WAIT error message' command halts the system, prints the error message and waits for operator intervention. Because of the uncertainties inherent in all real-world manipulations, WAVE offers a number of facilities.

SEARCH X,Y,Z sets up a box search in the x and y planes, starting in the x direction, with increments of 0.1 in figure 4. This pattern of initial guess followed by a sensor driven search, or some variation, is a standard technique in robot assembly programming.

Assembly operations

In a number of assembly operations, close fitting parts can be better mated with some of the degrees of freedom released, such that they are only balanced against gravity and acceleration forces. They will then comply to external imposed forces to prevent jamming.

FREE X,Y,Z gives translational compliance in the x and y directions.

SPIN 1.2
gives rotational compliance.

FORCE vector maintain the given force in the direction of the vector. To further ease the problems associated with close assembly:

WOBBLE 0.1

superimposes a 0.1m. sinusoidal oscillation on the hands' movement. These compliance and oscillatory modifiers are designed to reduce the incidence of close-fitting parts seizing together if force is applied at some angle not exactly perpendicular to the line of best fit.

The amount of processing required to convert these instructions into a form suitable to drive the arms motions is not trivial.

In this case the actual drive parameters are planned, using a model of the arms physical dimensions, possible motions and dynamic considerations in a time-shared

(continued on next page)
of these ideas will find their way on to the
work-shop floor. One may even, one day,
be able to program a robot in natural

Programming of mobile robots does not
need the same degree of transformational
arithmetic as manipulators, as they are, in
effect, only two-degree-of-

freedom devices. Because of this, there is
almost no need for highly-specialised
languages to control them—any
computer language will suffice so long as
the robot hardware is interfaced to the
software in some logical manner. Further-
more, the robot is seldom instructed in
terms of absolute co-ordinates, MOVETO
X,Y; but rather in terms of relative
movements, MOVEFORWARD 10 or

Figure 4. A box search.

GOLEFT UNTIL SENSOR3 > X.

Even when the algorithm functions in
absolute co-ordinate space the transformation
to relative motion, even if must be
planned, is straightforward. The
programming language LOGO has been used to
teach children about various concepts in
mathematics and computing using small,
two-wheeled, turtles which, with a
pen attached to their undersides can be
programmed to draw pictures on the
ground, according to programs the children
write—Papert, 1971a and 1971b, and
Papert and Solomon, 1971.

Less computation
As there is far less computation in
involved in determining the vehicle's actual
path, these languages can be interpreted.
Input text is scanned directly to perform
the actions, whereas WAVE had to pass
through a planning stage. The advantages of
easy testing, editing, rapid turnaround and
good diagnostics usually more than
outweigh the time overheads imposed by
interpretation. While the school children
will see only the simpler aspects of
LOGO, a full implementation of the
language can be used for complex A.I.
programming—bundy et al., 1978.

References
Amler AP and Popplestone R J (1974) Infor-
nating the position of bodies from speci-
fied spatial relationships in AISB Summer
Conference, July 1974 held at University of
Salzburg.
Bernorio M, Bertoni M, Dabnene A and
Somalaco M (1977). Programming a robot in
quasi-natural language. The Industrial
Robot, 4:3 (September 1977), pp. 132-140.
programming devices for teaching assembly,
inspection, material handling and produc-
tion tasks in 3CIR/6DISR paper D4, pp.
B4-33 to 42. International Fluidics Services
Ltd, Kempston, Bedford.
Brown J H (ed.) (1978) Artificial Intelligence:
An introduction. course. Edinburgh Universi-
Dragan P and Joffery M F (1976) Micro-
processor control and pneumatic drive of a
manipulator arm in 3CIR/6DISR paper D2,
pp. D2-9 to 20. International Fluidics Ser-
vice Ltd, Kempston, Bedford.
Finkel R H, Taylor R, Rutter R, Paul R, and
Fieldman J (1974), AL, a programming
system for automation. Stanford A I Lab.
Report AIM-423.
Gini G and Gini M (1978) Object description
with a manipulator in The Industrial Robot
6; (March 1978), pp. 32-35.
Hohn R E (1979) Application flexibility of a
computer controlled industrial robot. In
Industrial robotics, Vol I/Foundamentals.
Mitchigan: Society of manufacturing engi-
Hohn R E (1979) Robot programming in
Industrial robotics, Vol I/Foundamentals.
Mitchigan: Society of manufacturing engi-
Horn R M (1972) Robot manipulators, sta-
tics, and dynamics of two-dimensional
manipulators. In: Artificial Intelligence—
An MIT perspective Vol 2 (Winston P H and
MIT Research Institute (The APT long-range
program staff) (1967) APT Part Programming.
Kelly R R and Silvestro K C (1977) VI A
Visual instruction software system for pro-
gramming industrial robots in, The Indus-
Knuth D E (1968) The art of computer pro-
gramming, Vol I/Foundamental algorithms.
Reading, Massachusetts: Addison-Wesley
Publishing Co.
Lorero-Perez T (1979) A language for auto-
matic mechanical assembly in Artificial
Intelligence. An MIT perspective, Vol 2
Papert S (1971a). A computer laboratory for
elementary schools. M.I.T. A.I. Laboratory.
Artificial Intelligence Memo no. 246.
Papert S (1971b). Teaching children to be
mathematicians v. teaching about mathe-
matics. MIT Technology. Artificial
Intelligence Memo no. 249.
Papert S and Solomon C (1971). Twenty things
to do with a computer. MIT A1 Laboratory.
Artificial Intelligence Memo no. 248.
Paul R (1977) WAVE—A model-based lan-
guage for manipulator control in The
Paul R, Toed No S Y (1979) Human and
Robot task performance in Computer vision
and sensor-based robots. (Dodd G G and
Rossel L et al.) pp. 23-50. New York: Pienem
Rainier M H and Horn B K P (1978). Manipu-
lator control using the configuration space
method in The Industrial Robot 5:1 (June
Tidds J (1979). An investigatuio into the uses of
a microcomputer in the control of a
manipulator for tetraplegics in Engineering
and Medicine 9, pp. 193-200.