Computer Science 773

Robotics and Real-time Control

ROBOT PROGRAMMING

If a robot is a programmable machine, we must find some way of preparing programmes for it. There are (at least) two ways to construct the programmes, corresponding to two ways which can be used for conventional computer programming; interestingly, the easy way for computers is comparatively hard for robots, and vice versa.

It's perhaps worth emphasising that when we speak of robot programming, we don't include all the problems we mentioned in *STEP BY STEP TO ROBOT CONTROL*; we assume that someone else has kindly sorted those out for us, so all we have to worry about is the overall motion of the robot. Fortunately, this is a realistic assumption, if only because manufacturers have to provide controllers which can do all those low-level things if they want to sell their robots. Just now and then, market forces have their points.

The two ways are *on-line programming*, or *teaching*, in which you demonstrate directly by moving the robot about physically just what you want it to do, and *off-line programming*, where you sit down and write a programme in some programming language. Teaching is the easy way with robots, but the corresponding approach to conventional computing – sometimes called by names such as *programming by demonstration* – has met with rather little success, though comparatively recently script recorders such as are available for Applescript and Word macros have been quite successful.

TEACHING.

To teach a robot is quite simple. You position the robot in a sequence of configurations along a trajectory you wish it to repeat, and cause it to record each configuration by reading the robot's internal joint coordinate sensors. Once recorded, you can play it back again by directing the robot to move to the recorded positions one by one. All being well, this will result in motion which is close to what you want. (Most reasonably ambitious robot controllers can handle at least linear interpolation satisfactorily.) In case the performance is unsatisfactory, editing functions are provided; you can go to a particular position in the sequence and change it, or you can insert intermediate points for greater accuracy in tricky places.

There are different ways of recording the programme. You can manually manipulate the robot into the position you want, or you can drive it into position by operating its joints one by one, or by giving world coordinates of some sort. Many systems provide a *teaching pendant* – a small hand-held controller – for convenience in moving the joints and indicating the positions which are to be recorded.

If teaching will make the robot do what you want it to, it is a very good programming technique. As robots are very good at reproducing a position, it is very precise, and you don't need to do a lot of measurements to make it precise; it needs no special programming skills, so can be done by someone who knows about the task to be accomplished and is therefore likely to perform better than could be achieved by a programmer who wasn't familiar with the requirements.

As against that, there are some less satisfactory features. You can only teach a specific trajectory; some controllers can accommodate a library of trajectories, and can select from it according to conditions, but really flexible behaviour can't be learnt by rote. Even adapting the behaviour in obvious ways to conform to simple sensor input poses severe problems. It can also be very inconvenient to have to use the robot while programming; it is a fairly laborious process to teach a complicated trajectory and to make sure it's right, and taking your robot out of service for that length of time might have a serious effect on your production schedule. It is also unfortunately true that a taught programme is not usually transferable to a new robot of the original breaks down and must be replaced. Even if the two robots are ostensibly identical, the same

instructions will be unlikely to send them into exactly similar configurations, and if millimetres are important – which they often are – reprogramming is quite likely to be necessary.

Nevertheless, when it works, it works well, and it is a widely used technique for programming industrial robots where the requirement is for precise repetition of tasks which require complicated movements in space but no, or very few, decisions. This makes it a suitable technique for activities on assembly lines, which fit those specification very well, and where, once set up, long production runs can offset the time lost in programming.

OFF-LINE PROGRAMMING.

Writing a programme for a robot is much the same as writing a programme for a computer. You can use one of several fairly widely known robot languages of various sorts (which we shall discuss later), or you can write your programmes in a conventional language of your choice – nowadays, usually C or C++. If you choose the conventional language, you have to worry a bit more about getting the instructions to the robot, but most robot controllers now understand something like the standard MMS instructions, which they are happy to receive by conventional serial communications, so the fiddly bits aren't too fiddly.

If you want to use any significant amount of sensory input, programming is almost inevitable, for at least two reasons. First, the requirement for sensory input implies that the robot's behaviour is going to be quite variable, so you're unlikely to be able to use teaching; and, second, there is little or no standardisation in the way you can use sensory input, so it must all be programmed.

The advantages of off-line programming are fairly obvious. The main advantage is flexibility; the robot's motion can be made to adapt to circumstances – usually sensor inputs of one sort or another – comparatively straightforwardly, once the sensor inputs have been interpreted. A programmed robot at least has a chance of dealing with events which were not foreseen in detail when it was set up, because the programmes can be written to manage the robot's action in fairly general ways. The cost of off-line programming is not great, and production need not be stopped for long periods. Given the appropriate sensory equipment (which means, approximately, enough to implement position control), it is also possible to overcome the problems of non-transferability of programmes from robot to robot; provided that the body of the programme gets the configuration reasonably close to that required to do the work, final adjustments can then be made using sensor information as the robot goes about its task.

Disadvantages are also fairly obvious. If you're going to write programmes, you have to have programmers; the process engineers who are well equipped to teach a robot are not necessarily equally well equipped to write the programmes, and in any case if you're going to keep the plant running they'll be busy. A programme does need testing, and while you can do a certain amount by simulation you will almost certainly have to carry out the final tests on the plant itself – though this is likely to be less expensive in time than teaching the same actions. A programme is also more difficult to modify in case there are small changes to the specification.

WHICH IS BETTER ?

Both are, in their respective spheres. For simple repetitive operations, with limited variability, it's hard to beat teaching. At the other extreme, if you want complex behaviour which adapts extensively to circumstances perceived by sensors, you can't avoid programming.

Having said that, it is fair to add that manufacturing trends are away from very repetitive production of vast numbers of identical products. Even production lines are commonly built to incorporate some product variability, and robots which rely on

teaching might not fit into environments of this sort very well. Perhaps programming is the answer for the future, but teaching is likely to be with us for a long time yet.

BUT IN THE LONG RUN ..... ?

There are serious concerns that as we require our machines to be more and more intelligent (whatever we mean by that), and to be able to cope with more and more diverse circumstances, we shall find ourselves unable to write the programmes. Almost all industrial robots operate in worlds which are to some extent simplified; they are comparatively free of extraneous objects, everything that happens is governed by standard work practices and routines, and there are, on the whole, few surprises. In these environments, the most sophisticated robots are probably the automatically guided vehicles used in some plants to move products and equipment around. Advanced versions are able to find their way around the plant, avoid obvious small stationary obstacles, and not run over people. And that's about it.

Suppose you want to build a robot which will deliver your completed products to your customers. It will have to behave sensibly on public roads, understand traffic regulations (and adapt when they change), avoid idiot pedestrians who run in front of them, work out when to try another route, answer questions from policemen ....

How do you write the programme for that ? Do you have to write a separate procedure to cope with every eventuality ? The sheer volume of code would be unmanageable. Do you try to programme your robot to behave like a human ? How do you do that ?

No one knows the answer. There are fairly convincing arguments that it really is beyond our capabilities; if so, then our best prospect for building such intelligent robots might be to try to make them in such a way that they can learn, much as we do.

But no one knows how to do that either.

Alan Creak,
April, 1998.