Computer Science 773

Robotics and Real-time Control

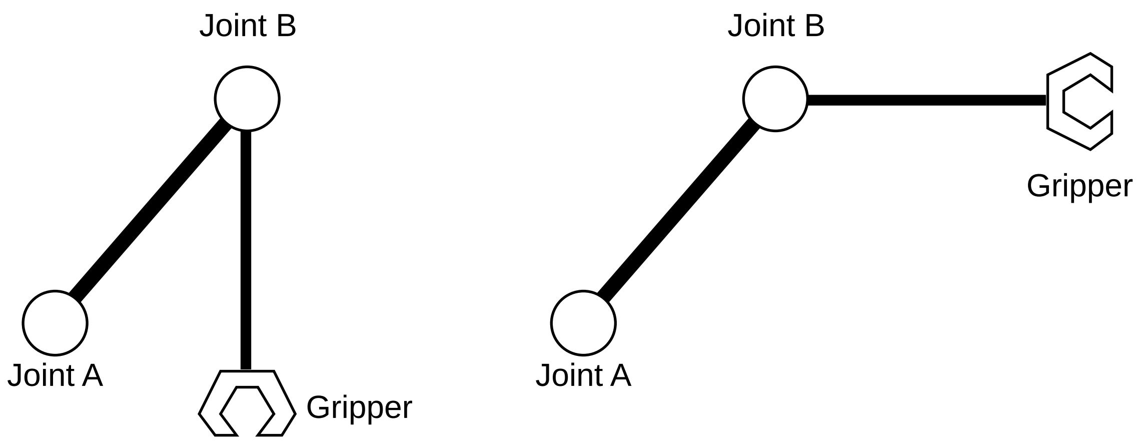
CONTROLLING ROBOTS

If there were nothing about controlling robots which didn't happen in other machines, there wouldn't be much point treating them as a separate topic. A robot certainly is a machine, so we'll expect that some of the methods we've met in ordinary control systems will appear again. Also, though, we have seen that robots are rather special in that they are general-purpose machines, so there might be control functions which operate at this level.

LOW-LEVEL CONTROL.

At the lowest level of interest to us, a robot is a collection of motors, each of which must be controlled so that its output position (shaft rotation or distance) and velocity follow paths determined to produce the required motion in the end-effector.

At this level, control can be exercised by a simple controller with a variable set point (the desired position or velocity). A closed-loop controller is usually used; open-loop controllers are usually not useful, because the environment of most of the individual motors changes from moment to moment as the configuration of the robot changes, so there is no simple way to work out the control signal required for an open-loop signal. Consider, for example, a simple two-joint robot, as shown in the diagram.



Although the local configuration of joint A is the same in both cases, significantly more torque will be needed for the same acceleration in the left-hand configuration than in the right-hand configuration, as the mass is concentrated closer to the centre of rotation in the left-hand case. As an second example, suppose that it is required to move from the left-hand to the right-hand configuration by keeping joint A fixed while rotating the gripper link around joint B; clearly, the joint B motor must be used to rotate the gripper link – but in doing so, it applies a torque to the A-B link, so the joint A motor must be activated in the opposite sense to keep the link still.

There is one special case (perhaps there are others, but I don't know about them) where open-loop control is possible : if the robot is built with stepper motors. The special feature of a stepper motor is that you know what the motor is doing without requiring feedback. That works for small robots with comparatively light loads, but for more demanding tasks stepper motors are a very expensive way of getting the required torque, and DC motors (or, for the largest machines, hydraulic actuators) are more common.

An additional potential hazard with a stepper motor is that if it happens to slip a step at some point there is no way of finding out, so all actions after that event will be wrong.

HIGH-LEVEL CONTROL.

The most obvious requirement for control at a higher level follows from the preceding discussion; if, as is usual, the end-effector is required to follow some defined trajectory in space, the actions of the separate motors must be coordinated. At the simplest level, a sequence of set points must be sent to all the motors from time to time, where the intervals between set points are short enough to guarantee that the overall motion is satisfactorily close to the desired trajectory.

In practice, this very simple-minded approach is only satisfactory for rather slow imprecise operation, and is not very satisfactory for that. Repeatedly altering the set point for a simple controller is likely to result in uneven motion with rapidly alternating periods of acceleration and deceleration, which is wasteful of power and doesn't do the motors any good at all. For better and smoother motion, and for accurate faster motion, it is better to use more sophisticated control at the higher level, so that the complete motion of the whole robot is planned beforehand, and required positions and velocities specified for all actuators at intervals. This observation is the beginning of a battle with the physics and geometry of robots which is developed in further detail in the sheet *STEP BY STEP TO ROBOT CONTROL*.

Above this level, something has to produce a specification of the required robot motion. This is where the notion of programming the robot fits in. In principle, one could write the programme for a robot as a sequence of joint coordinates with associated time intervals, but this would be intolerably tedious. At least two other ways of programming robots have therefore been developed.

The simpler way for the operator is to *teach* the robot the required sequence of movements; the operator moves the robot through a sequence of configurations and instructs it to remember a certain set of these. Later, the same set can be "played back" when instructed, so that the robot covers the same trajectory again, reliably and repeatedly. This is a very effective way to provide a programme for a simple, invariant, repetitive task – which is the sort of task which many robots perform. This is sometimes described as *on-line programming*.

If you want your robot to behave in ways which are not simple, invariant, and repetitive, teaching will not get you very far. To deal with a broader range of events (the robot must pick up components which arrive in different orientations, mobile robots must avoid obstacles, etc.) you have to give the robot instructions from which it can work out its behaviour, not a rigid specification of the behaviour itself. There have been experiments in teaching such higher-level notions, and research continues, but in practice it is difficult so far to attain any degree of sophisticated interaction with the environment without some form of *off-line programming*, which is programming in the sense it usually carries with computers : some sort of programming language is used to encode instructions which the robot follows. This topic is developed further in the sheet *ROBOT PROGRAMMING LANGUAGES*.

SPACE.

One factor which differentiates robot programming from other sorts is its obsession with moving about in space. Just as timing problems dominate the practice of real-time programming, so spatial problems are overwhelmingly important in robot programming.

This shouldn't be a big surprise, because it follows fairly directly from the nature of robots as general-purpose machines. Much of real-time control is concerned with moving things about in space, but in other parts of the subject it is usually possible to simplify space, and we usually do just that. A conveyor belt reduces space to one dimension, and we can put special position sensors at points along the belt so that we don't even have to worry about the single dimension that's left; in using machine tools, it is common to fit parts for machining into a standard jig which removes any uncertainly about their positions; moving parts of machines rarely move freely, but are constrained by axles or guides or rails or linkages to stick to predetermined paths with (if it's absolutely unavoidable) one degree of freedom.

You can't do that with a general-purpose machine, if it is to be suitable for general purposes. Robots must therefore be able to cope with the six degrees of freedom (three for position, three for orientation) of objects in ordinary space, and in programming them one must be able to handle all the implied geometry. One reason for the comparative simplicity of teaching methods is that you bypass the geometry, but any sort of off-line programming must have provision for more or less elaborate geometrical and trigonometrical calculations.

Alan Creak,
April, 1997.