entirely — typically revolute co-ordinates. So the robot must be able to solve some tricky geometrical problems in order to work efficiently. And, in the case of industrial robots that must move objects weighing several hundred kilograms over a large distance, the saving in time and energy by choosing the best route can be considerable.

Another problem facing the point-to-point robot is the dynamics of the arm itself. Move your own arm to pick up an object and you will find that it accelerates slowly away from its original position until it reaches maximum speed, then decelerates until it comes to a smooth stop at its final position. The advantages of a robot arm that does this are considerable. Many such arms move at a constant speed, accelerating almost immediately to maximum velocity and stopping dead at the end of the movement sequence. This places strain on the arm itself and requires more power than an arm which accelerates and decelerates smoothly. It also means that the arm may not move as quickly as it could otherwise do and, if at the end of the sequence the robot were required to pick up a delicate object, even a slight displacement of the object could result in the arm hitting it with considerable force. So the robot needs to work out an optimum speed as well as an ideal path to follow.

ROBOT CHOREOGRAPHY

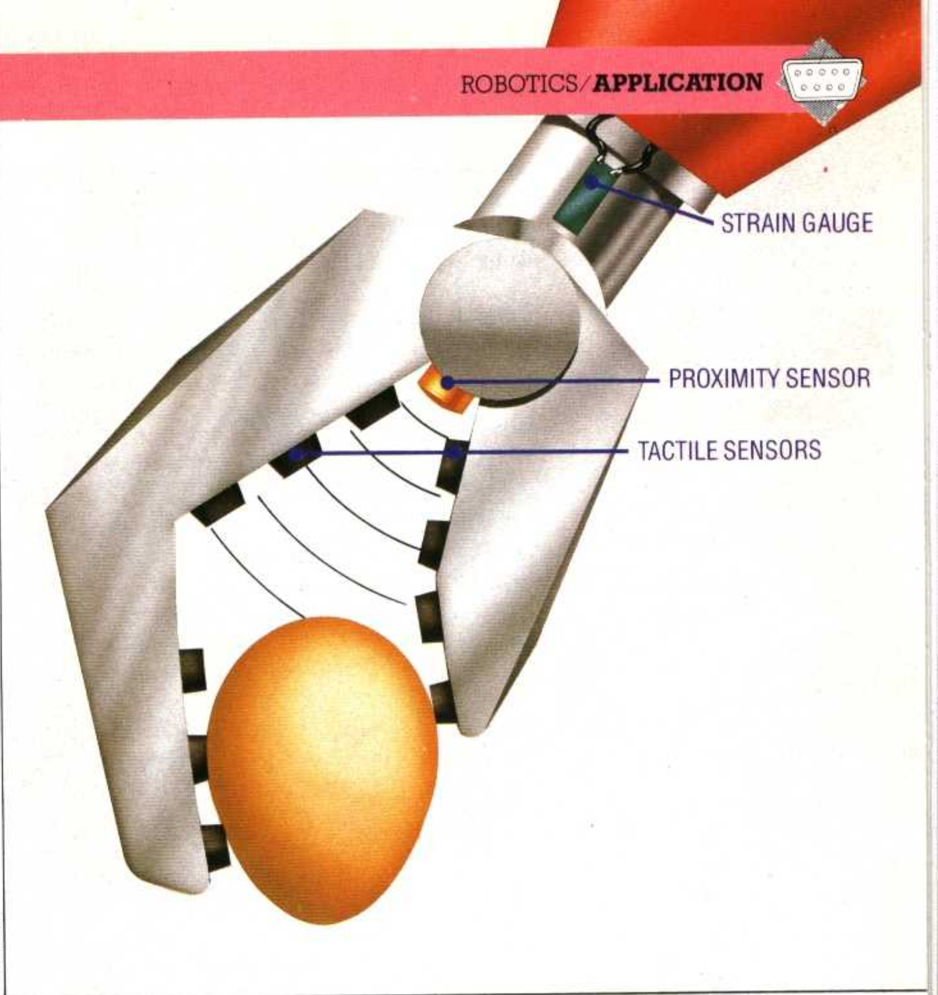
Even after the arm has been programmed to follow a precise set of movements, it may turn out on playback that these movements were not quite what was required. This may be because of human error, because the nature of the task has changed slightly, or because, if the robot is working in conjunction with other arms, the arms may follow their own paths and collide with each other. (Avoiding this latter problem is known as 'robot choreography'.) So a method of editing the sequence is required. This can be achieved by storing the movements as a linked list, in which each position is stored and followed by the address at which the next postion is to be found. In the initial training session, this address will be the address of the next position in the list. If the sequence needs to be edited, the arm could be moved to the position at which corrections need to be made, stopped, and a new sequence inserted.

Another common method of making arms move intelligently is to use a series of programmed instructions stored in the computer. Typically, each robot has its own programming method and uses a different programming 'language' to control movement, but in general a language is required that enables the programmer to use LOGO-like commands to specify movement in three dimensions, with added instructions for wrist and end effector movement, such as 'pick up' or 'put down'.

The problem is similar to training a point-to-point robot, and many factors need to be taken into consideration. For instance, if a robot arm is to move forward 10 units then the obvious method would be to alter the shoulder joint so that the arm can reach further forward. However, this would cause the arm to move upwards in an arc, and so this must be corrected by a downward movement in the elbow joint. From this it can be seen that the instruction for just one simple movement must be translated into two distinct sets of instructions, working on two separate joints.

Other problems can arise when the robot is required to pick up an object. However well the arm is positioned, it is difficult to ensure that it is in exactly the right place to pick up the object, particularly if that object is of an asymmetrical shape. An 'intelligent' hand is therefore needed — this must sense the presence or absence of the object, the distance of the object from the hand, and the force exerted by the hand when it tries to pick the object up. These problems may be tackled by equipping the hand with a range of proximity, tactile and force sensors, which provide feedback that enables the controlling computer to make any necessary corrections.

If all these problems are considered, we can see that it is possible to construct a robot arm that shows a relatively high degree of 'intelligence'. However, as yet no arm can be designed to, say, bowl a cricket ball accurately. This is because the arm's intelligence alone is not enough. The robot must also know the state of the pitch, the position of the batsman, the wind strength and direction, and a host of other variable conditions. Then, of course, it will need to be able to work out the very complex equations involved in sending a projectile through the air. For such tasks, much more than just an intelligent arm is required.



Gently Does It
Picking up an egg is a searching test of the robot arm's sensors and feedback control mechanisms. The gripper's proximity sensor must check that the egg is close enough to grasp, then the fingers can start to close until the touch sensors indicate contact with the egg. The output of the touch sensors must now be checked against that of the proximity sensor as the fingers close and the arm begins to lift. A sudden decrease in proximity shows that the egg is slipping, so the fingers must tighten until a preset grip-force limit is reached or until sudden decrease in grip-force shows that the eggshell is distorting prior to cracking