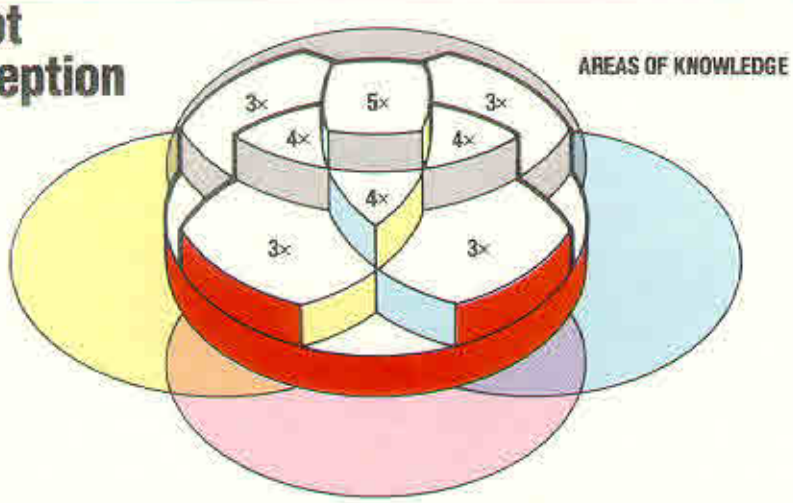
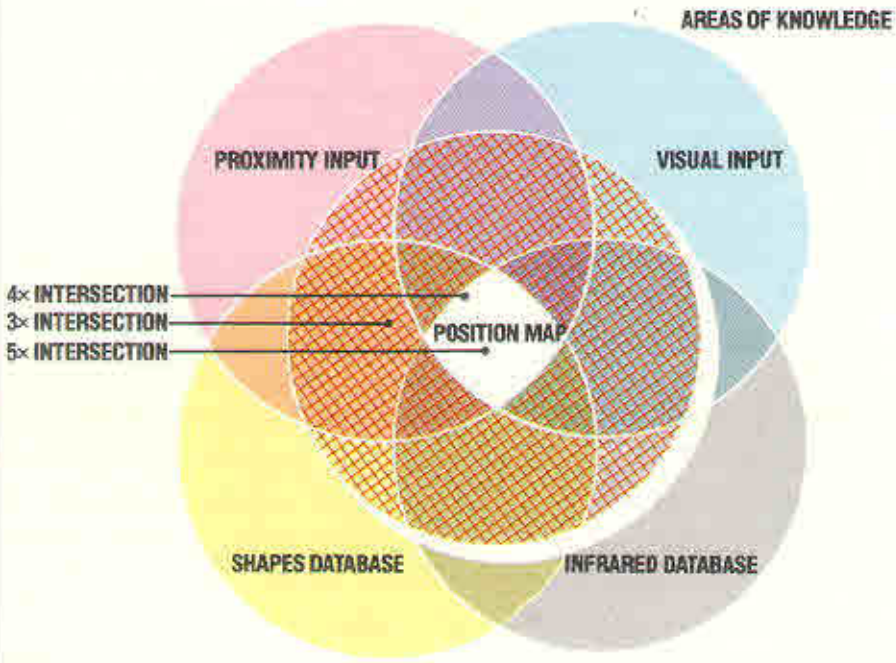
Robot Perception





In the illustration, with five areas of knowledge, there is one intersection of all five circles, four different intersections of four circles, and so on. The robot checks any object in view against the five-times intersection, then against each of the four-times intersections, and so on down the levels of intersection until a match is achieved; the higher the level of intersection, the more trustworthy the inference

KEVIN JONES

Intersection Interface
Robots usually have several sources of knowledge available to them: some are their pre-programmed databases (of common object silhouettes and infrared signatures, for example) some are experiential databases (such as the robot's current map of its surroundings), and some are sensory input channels (such as proximity and shape of objects). In the ideal case — when the robot 'knows' its location and 'understands' its surroundings — then the intersection of these areas of knowledge should uniquely identify any object in the robot's view

used by the maze-solving robot (see page 772) that uses sensors to work out the position of walls in the maze, building up an internal, two-dimensional map as it progresses. If we extend this thinking to an arm, the map must be three-dimensional. Add vision to the robot and the three-dimensional map immediately acquires colours, variations in brightness, and patterns that no touch sensor could have detected. With some measure of speech recognition, the robot adds spoken information to its model of the world.

The problem confronting robot designers who are trying to enable the robot to make sense of its environment is that the world is not static and is constantly changing. Therefore, the robot also needs to be equipped with some means of allowing for such change.

If we consider a robot that is programmed to perform some simple task like stacking bricks, the extent of this problem becomes clear. If the bricks are of unequal size, they must be positioned one on top of another very carefully and if the centre of gravity of each brick moves outside the base area of the bricks the whole pile will topple over. But what knowledge does the robot have of the laws of gravity? And if the pile does topple over, would it understand what had happened and take the necessary action?

PROBLEM SOLVING

There are two main approaches to this problem. The first is to program the robot with data that includes a prescribed course of action for any eventuality. This would obviously limit the robot's understanding to certain clearly-defined tasks. The brick-stacking robot would thus be programmed with instructions to ensure that each brick was placed exactly on top of the brick underneath, with the centre of gravity of one directly above the centre of gravity of another.

The second approach is that advocated by those who argue that the only way a robot can ever understand its environment is by learning about it for itself. This is a field of computer science known as learning or *heuristics*. With this approach, the robot is programmed to perform a particular task and is provided with feedback, either from a person or from its own sensor, which will tell it how well it has done. With reference to this feedback, the robot will modify its own internal program — its own model of the world — in order to improve its performance and build up a 'reference library' that will help it cope with future tasks. The maze-solving robot acts in this way when it tries to find the best way out of a maze. By using its sensors, it can detect any blind alleys and will then take the necessary action by retreating back up the path and trying a new one. Unfortunately, there is no single learning program that may be used whenever a robot needs to learn a task.

Once the robot has 'learnt' the necessary lesson, it needs to store whatever this is as a computer program. This task is known as *knowledge representation*. Traditionally, the robot's knowledge may be stored as the lines of code that comprise an ordinary computer program. But artificial intelligence techniques have led to other approaches. There is a wide range of techniques in use, but the most common include *production rules*, *semantic nets* and *frames*.

Production rules are in the form of IF. . .THEN constructs and are simple statements of fact. So, a robot may store its knowledge in the form: IF there is a brick wall in front of you THEN you cannot go forwards. There can be a whole sequence of rules of this type — they have the advantage of being easy to write and easy to understand for the programmer who is writing the program. The snag is that the robot has to understand them too — it needs what is often called an *inference engine* to be able to interpret these rules into a course of action. Programs using the production rules may be written in a conventional language, such as BASIC, but are more commonly written in a *declarative* language, such as PROLOG, which is better designed to handle this sort of knowledge. This is because, unlike the traditional languages, declarative languages do not execute their instructions one at