SUBSUMPTION ARCHITECTURE FOR THE CONTROL OF ROBOTS

Daniel Toal[1], Colin Flanagan[1], Caimin Jones[1] and Bob Strunz[2].

1. Department of Electronic & Computer Engineering
2. Information Technology Department.
University of Limerick.

ABSTRACT

This paper describes ongoing investigations into the subsumption control of robots. Subsumption architectures are a relatively new and simple approach to the control of robot systems. The technique has been applied to the control of experimental mobile robots. Some of the potential benefits of subsumption control over classical robot control techniques may prove useful in the context of industrial robotics.

With subsumption control the robot is not controlled by a powerful reasoning system, but rather by a set of simple processes. Each of these processes is designed to provide the robot with a simple competence, for example, the ability to move away from an obstacle, to move around an area in a random fashion, or to explore the robot's environment. The processes compete for control of the robot at any given time, according to a varying priority scheme which is mediated, *inter alia*, by external sensor input. The subsumption approach to robot control represents an interesting alternative to classical artificial intelligence techniques, but remains unproven at this time.

A small mobile robot testbed has been built to investigate the feasibility of subsumption control for robots. Three simple robot processes have been developed. These are a random motion ability, a collision avoidance competence and a light seeking/ following ability.

Implementing primitive abilities such as a wandering behaviour and collision avoidance on both mobile robots and robot arms has potential applications in industrial robotics. Using a subsumption architecture could allow supervisory control to simply deal with task oriented decision making leaving lower level reaction to unexpected or uncontrolled environment conditions to lower level subsumption abilities.

1. INTRODUCTION

The real world is a complex, unstructured environment. Robots which are designed to operate in the real world must be able to operate in situations which their designers only vaguely envisaged and must have the ability to respond appropriately and quickly to unexpected events.

The "classical" artificial intelligence approach to interacting with an environment is to divide the task into a number of major subsystems, typically: (1) perception, (2) world modelling, (3) planning, (4) task execution, and (5) motor control. The perceptual subsystem handles the sensing devices connected to the robot. The modelling subsystem converts the sensor input into a description of where the robot is in relation to its internal model of the environment. The planning subsystem attempts to work out how it will achieve its goals given the current world state. Task execution breaks down the plan into detailed motion commands and finally the motor control causes these commands to be executed.

Each of these subsystems is a complex program, and all have to work together perfectly for the robot to operate at all. Some of the subsystems are extremely complex, e.g., the tasks of perception and world modelling. Currently it is only possible to design such subsystems for structured environments. The noisy and random environment of the real world overwhelms them. In particular, as the complexity of the environment increases, the time needed to perceive, model and plan about the world increases exponentially.

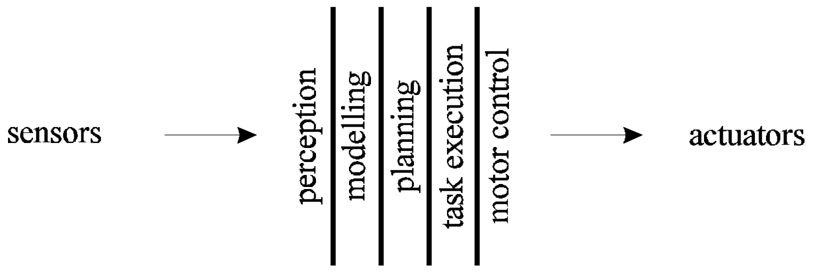

Figure 1. The Classical Artificial Intelligence Approach to Robot Control

This "classical" approach to robot control can be though of as a type of "vertical" division of the control problem into functional units, each of which is responsible for a well-defined task which is incapable of controlling the robot by itself. An important feature of this approach to robot control is that the world perceived by the robot must be reduced to some sort of "symbolic" description which can then be reasoned about by the planning subsystem using the classical artificial intelligence approach based on state-space search and tree-pruning. However, this task of "grounding" symbols in aspects of the real world has been, and remains, one of the most difficult and elusive goals of artificial intelligence.

The subsumption approach to robot control is an alternative to the "classical" approach. It is a minimalist approach. Instead of having a number of complex individual "vertical" tasks, the subsumption approach tackles the control problem by thinking of it as a number of "horizontally" arranged "layers". Each layer is designed to implement a "competence", i.e., the ability to display a behaviour. Thus, each layer is fully capable of controlling the robot by itself, albeit in a possibly simple-minded way.

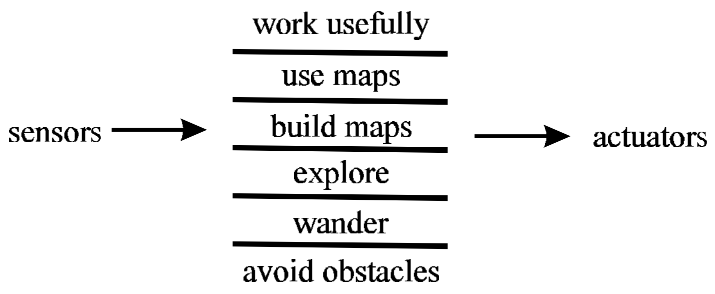

Figure 2. Subsumption Architecture

Thus, each of the "horizontal layers" contains elements of all the "vertical" tasks found in a classical system, usually implemented very simply. Each layer implements one behaviour, such as the ability to move away from an obstacle, to move around an area in a random fashion, or to explore the robot's environment. Thus, each process need not tackle the whole complex problem of perceiving everything in its environment, or planning the whole business of getting the robot from A to B, instead, it only performs that part of perception, planning, etc., which is appropriate to the task it has to perform. As an example, a process which is designed to implement obstacle avoidance need only be able to move away from an obstacle "noticed" by the robot's sensors

A "classical" subsumption architecture, as described by Brooks [1] consists of a set of complete robot control systems, each of which achieves a particular level of "competence". Brooks has defined eight such layers of competence which he has labelled layers 0 to 7. Of these only layers 0 to 2 have been implemented on a robot:

The layer 0 control system provides the robot with the ability to avoid contact with objects (whether these are moving or stationary).

The layer 1 system allows the robot to wander around aimlessly without hitting things.

Layer 2 endows the robot with the ability to "explore" the world by seeing places in the distance using its sensor array and heading for them if they appear to be reachable.

Layers 3 through 7 add much more complex behaviour such as the ability to map the environment, formulate plans about it and reason about the state of the world. The team headed by Brooks has made some initial investigations of how such behaviours can be implemented in a robot, but it is not yet clear how successful these will be in the long term.

Industrial robots usually operate in a more controlled environment than experimental mobile robots. Careful design of the industrial robot's environment and design for manufacture of the piece parts the robot works on greatly simplify the robot control problem. Working in a controlled structured environment significantly reduces the perception and world modelling sub-elements of the control task as described above. Never-the-less inclusion of advanced sensory systems such as vision systems does imply an increase in the complexity of control in the areas of perception, interpretation and world modelling. Also, flexibility is one of the most important attributes of robotics over other forms of automation. This flexibility implies uncertainty in the robot environment. For these reasons, subsumption control may therefore have a useful role to play in certain applications of industrial robotics.

2. OUR AIMS

In this paper we describe work being undertaken at the University of Limerick on mobile robots employing subsumption architectures. Our initial aim in this work was simply to develop a robot equivalent in abilities to those discussed in Brook's early work. Initial work on the project, including building a mobile robot hardware base and developing some primitive subsumption processes on this robot is described by Flanagan et. al. [2]. Further work has been carried out on the development, testing & integration of sensory sub-systems using ultrasonics and light seeking arrays and implementing simple subsumption capabilities employing these sensors.

3. ROBOT DESIGN

For a detailed description of the mobile robot see Flanagan et. al, [2]. The robot was designed primarily to perform simple motion tasks under the direct control of an on-board computer. The on board computer receives motion commands from an external PC. Status and sensory information is fed back to the PC. Design flexibility allows for easy modification and expansion

3.1 Physical Hardware

The robot has two independent drive wheels mounted beneath the base plate, one on either side providing locomotion. Directional control is achieved by driving both wheels together, driving one at a time or by driving them in opposite directions. The robot can thus drive in straight lines, rotate on the spot, or turn with one wheel stationary. Move distance, speed and acceleration are programmable.

Two 2", 2 phase standard hybrid stepper motors drive the wheels using a rubber belt and pulley transmission system providing a 5:1 reduction ratio. The robot was designed to achieve a target speed of 1 m / s and an acceleration rate of 1 m / s^2. Both of these design targets were exceeded.

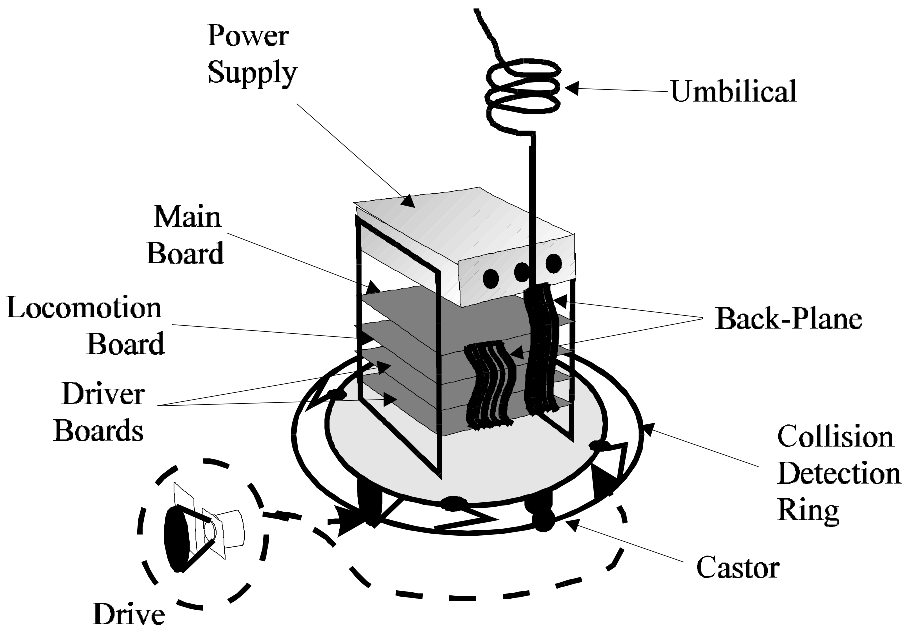

Figure 3. The Robot

The robot has a simple collision detection system consisting of a ring connected to four micro switches which are interfaced to the on-board controller. The robot also has a more sophisticated collision detection and obstacle avoidance systems which employs ultra sonic transmitters and detectors, This sub system is described in greater detail below.

3.2 Software - Robot Control

Robot control has been split into on-board low-level control and high-level control hosted on an external PC. Simple software has been written on both the on-board controller and supervisory PC. The onboard software includes motion primitives, input / output & sensory routines and serial communications software. Implementation of the sub-sumption control uses a high level command language run on the PC, which is at an early stage of development.

The above approach of separating the low level on-board control primitives and the higher level task oriented control is very similar to that used by Brooks in designing ALVINN, an early robot built at the MIT Mobot lab. It has the advantage that the subsumption control algorithms are easy to change and can run on a powerful machine. Ideally, of course, the entire control system would be hosted on the robot, along with all the electronics and the power supply. We deemed this impractical at this stage and regarded flexibility as being of primary importance.

3.3 Low-Level Control (on-board)

The on-board controller consists of a Z80 based main board, a locomotion interface card, and two power driver boards for the motors. All main board control, address and data lines are available to other robot boards via a back plane. At present the main board has 16Kbytes of on board memory which is expandable to 64K. All necessary control and enable signals for the extra memory chips are present on the board.

Stepper motors have limited start /stop regions such that for most working speeds it is necessary to start the motor at a low speed and accelerate up to the desired speed. It is also necessary to decelerate the motor back to a low speed before stopping to avoid overshoot. This speed profiling task is implemented in digital hardware on the robot, freeing up the CPU for other tasks and for future expansion. The locomotion board provides hardware control of move distance, final speed and acceleration / deceleration of the stepper motors beyond motor start / stop speed limits using a buffered clock arrangement.

To realise a movement the Z80 simply transfers the necessary speed and distance constants to the locomotion board. The acceleration rate and deceleration provided by the buffered clock are also programmable from the CPU. Once set up, however, the acceleration rate is rarely changed unless required by the robot environment. An example would be if the robot encounters an incline. A buffered clock monitoring capability has been included such that the CPU can interrogate for current speed and move distance. This feature is important under collision conditions for monitoring current location.

The CPU can enable both driver boards such that the robot moves in a straight line. Each motor can be reversed by reversing the phase sequences applied to the motors. This allows the robot to turn on the spot, either clockwise or anti-clockwise, again under CPU control. By disabling one drive the robot can turn, with one wheel stationary. Combinations of straight line, turning and rotating motions will allow the robot to be controlled to negotiate its environment. Differential motor speed control, allowing the robot follow an arc, has not yet been provided. This could however be provided without much difficulty. The currently implemented manoeuvring capabilities are sufficient to begin.

Motor driver chips and other power electronics are mounted on separate boards in order to separate the high power analogue circuitry from the microprocessor and logic boards. Full stepping of the motors with both phases on is the stepping mode used as it gives the greatest torque output. The 200 steps per revolution were more than adequate. 1 mm of travel corresponds to 5.3 motor steps. Motor control was realised with SGS-Thomson L297 and L298 controller and driver chip pairs.

A bumper ring connected to four micro switches is used for simple collision detection. Depending on the combination of switches that are closed the CPU can determine the sector or quadrant of collision. A /COLLISION line is used to stop the motors, via hardware, immediately a switch is closed. This feature can be disabled from software with an /OVERRIDE output. This feature can enable the robot to push its way through light obstacles. Ultrasonic obstacle detection and the light seeking capability are described below in section 4.

3.4 Robot to PC Interface

Communication between the robot and the PC running the high-level control software is via an RS232C link. This low bandwidth link is adequate for sending simple commands to the robot and receiving low-bandwidth data from it. Currently the robot only provides simple obstacle detection, using ultra sonics and micro-switches, and the light sensor differential signal. These represent a low bandwidth input data stream to the PC. When the robot is extended to include higher-bandwidth data sources, these will require their own links back to the PC. This is again a conscious design decision. The Z80 has enough to do controlling the motors. Higher level sensory capabilities will ride on the robot as self contained systems and will not interfere with the basic control. This is a good example of subsumption in practice.

4. ULTRA-SONIC DETECTION AND LIGHT SEEKING ABILITIES

An ultrasonic ranging / obstacle avoidance system and a light seeking ability have been designed and implemented on the mobile robot and are used within certain subsumption processes.

4.1 Ultra-Sonic Detection

This system enables the detection of obstacles in the vacinity of the robot prior to collision. However the collision detection ring still acts as a fail safe back-up system to the ultrasonics. Thus far the ultra-sonic 'vision' capability has been implemented as a look ahead feature with transmitters and receivers covering the area directly in front of the robot. This 'vision' capability can easily be expanded to include lateral and reverse vision although this will be done with less coverage and definition.

Two techniques were researched with regard to ultrasonic detection. A pulsed 'time of flight' echo system similar in operation to radar systems was successfully developed and gives information relating to absence / presence of an obstacle and also obstacle range information. Velocity of obstacles relative to the motion of the robot can easily be determined by differentiating the range values or signal.

A second technique was investigated but has not yet yielded a reliable solution. If the intensity of the ultrasonic echo can be measured it can usefully be used to generate priority of required response information. The intensity of reflected energy from a target/obstacle falls off with R^4 (R - range). Put another way, if the robot is approaching an obstacle or if a moving 'obstacle' is approaching the robot, the intensity signal from the ultra-sonic receiver will squeal as the range decreases. Thus the rate of change of the intensity signal can be used as a powerful priority measure. The rate of change of intensity is expected to give a better warning system than simply measuring the intensity of reflected energy. Intensity of reflective energy varies greatly with many factors, e.g. material of target, surface finish, dimensions of target, etc.

4.2 Light Seeking Ability

A light seeking capability has been implemented on the robot using an array of light sensitive resistors as detectors combined with signal conditioning and amplification. As currently configured this sensory system generates signals which are used to determine the direction of a light source relative to the direction in which the robot is facing. The light seeking control process implemented on the robot simply drives the robot to align it with the light source.

5. HIGH-LEVEL (SUBSUMPTION) CONTROL

The high-level control of the robot is vested in a "subsumption-like" software architecture running on a host PC. Our implementation of subsumption differs somewhat from that proposed in the original literature.

Brooks originally proposed that a subsumption architecture be comprised of a hierarchy of controller layers where each layer is capable of overriding all inferior layers at any time and taking control of the robot for as long as it wishes. It then relinquishes control to whatever inferior layer was previously in command of the robot. The layer which "subsumes" control of the robot at any point need have no knowledge of what is currently controlling the robot at that point, and similarly, the "usurped" module need have no information about the "subsumer".

We propose two weaknesses with this scheme:

(i) In practice it has been found that this very loose coupling between layers is not sustainable. Layers often need to pass information back and forth, but "pure" subsumption control only allows this on an ad-hoc basis. We feel that a more structured approach is needed here.

(ii) The division between layers is quite rigid. Thus, if a layer 2 competence decides that it wants to control the robot, a layer 0 competence cannot resist it. This is potentially hazardous if the layer 0 competence has abilities that the layer 2 competence neglects. If layer 2 does not understand the behaviour required to retreat in front of a charging bull, but layer 0 does, layer 0 cannot stop disaster from occurring! This leads to the requirement that each layer implement the full competence of all lower layers in addition to its own tasks. We feel that such an approach is wasteful of resources. Yet the typical solution adopted by the Mobot lab machines is for higher layers to "grope around" in the internals of their subordinates. This is also undesirable from a software engineering standpoint. The difficulty seems to arise from the rigid and early decision about what competencies belong "above" others.

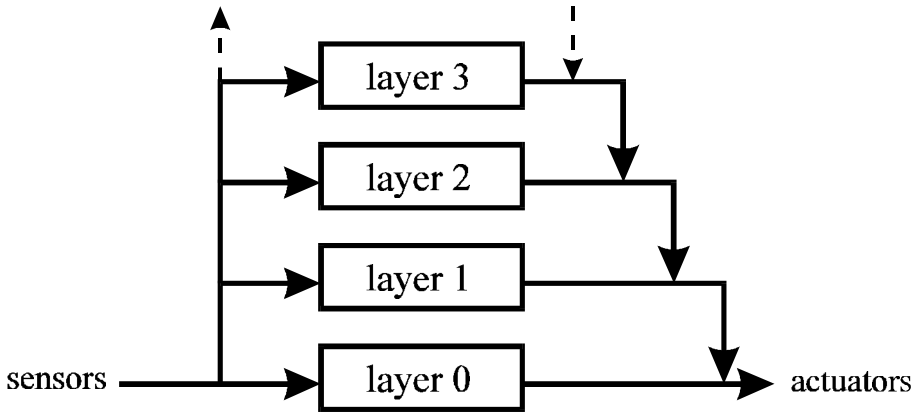

Figure 4. Brook's Subsumption Architecture

Our alternative approach is based upon the following ideas:

(i) We substitute a set of independent processes for the layers of the Brooks architecture. Each process is responsible for implementing a task which represents some behaviour of the sort handled by a layer in the Brooks architecture, or possibly a simpler task.

(ii) We maintain a global "blackboard" data structure where processes may place information they consider "interesting" and read the postings placed by other processes. This area is quite unstructured, processes may place whatever they like there (with obvious limitations imposed by memory size, etc.). The only real requirement of the area is that every piece of data be clearly stamped to reveal its originator and time of generation.

(iii) We allow the processes to "compete" for control of the robot at any instant, according to a varying priority scheme. Urgent events in the external world (e.g., sensor input telling the robot that it has just crashed into a wall) generate high priorities for processes associated with them, such as the obstacle avoidance task. We also propose that priorities be time varying, so no one process has a chance to "hog" control of the robot. We expect the robot to demonstrate an "attention span" due to decaying priorities.

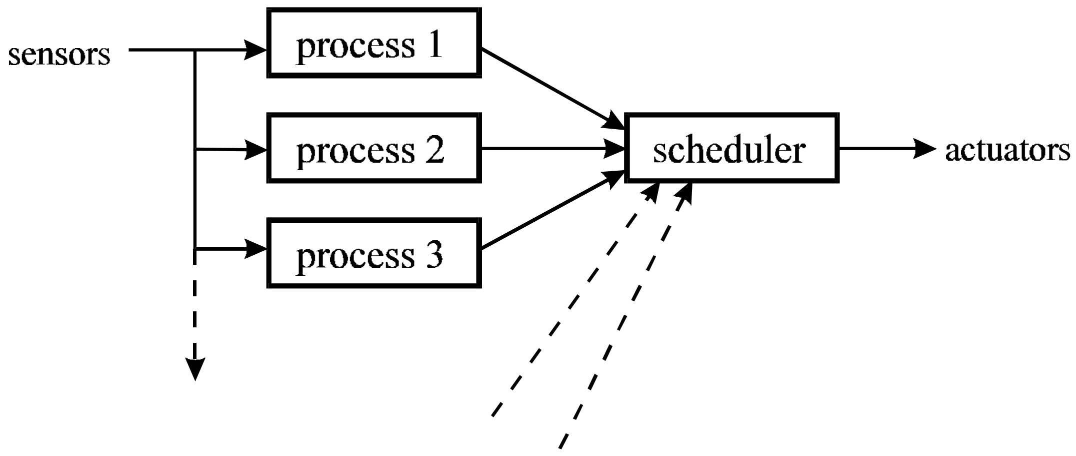

Figure 5. Modified Subsumption Architect

This scheme has potentially two weaknesses in comparison with the Brooks approach. It relies on a global data structure and it requires a centralised arbitrator to decide which process controls the machine at a given moment. We believe that neither of these is a serious weakness, but only experiment can confirm or deny this hypothesis. On the other hand, we believe that the flexible priority scheme discussed here is an advance on the basic Brooks model. It is interesting to note that a similar idea has appeared in the more recent work from the Mobot laboratory. In any event, we intend to contrast the behaviour of the robot using our approach and Brooks' to see what the relative advantages of the two are in practice.

In the Brooks architecture each layer is programmed as a set of asynchronous processes which communicate via message passing. These processes are extremely simple and are described as augmented finite state machines (AFSMs). A special language (the "behaviour language", a subset of Lisp) has been defined by the Mobot lab researchers in order to design these AFSMs, and compilers have been written to convert behaviour language descriptions to networks of AFSMs in various machine codes. We intend to work at a different level of detail and treat each of our competing processes as a unit instead of as a network of much simpler entities.

6. POSSIBLE AREAS OF APPLICATION OF SUBSUMPTION IN MANUFACTURING

Current industrial robots are designed for well structured environments. The noisy and random environment of the real world unless properly controlled would overwhelm them. However, application flexibility is one of the most important attributes of robotics over other forms of industrial automation. This flexibility implies some uncertainty in the robot environment. Subsumption control may therefore have a useful role to play in certain applications of industrial robotics.

Subsumption, as described above, could prove useful in the control of AGV's. An AGV that encounters an obstacle may well bypass the hold-up through subsumption. Alternatively, an AGV could have a low priority cleaning process which is enabled whenever the more common transport tasks are unwanted A robot arm could be programmed off-line to perform a certain function without regard to crashing into hardware, parts and fixtures in a cell. If such a robot arm was equipped with sufficient sensors, path modification for collision avoidance could be achieved during a first or dry run using subsumption. Modified paths could be recorded and incorporated in future program runs.

Such scenarios would simplify the off-line task-level programming of robots leaving unforeseen and varying implementation details and reaction to unexpected or uncontrolled environment conditions to the run-time controller and subsumption abilities.

7. RESULTS & CONCLUSIONS

At this point in time the robot has been built and tested. Construction of the initial version of the PC-hosted control software is well underway. The project is in its early stages, so portions of the subsumption mechanism have been implemented and tested at the time of writing, but this testing has not been rigorous.

We have thus far implemented a basic controller containing a number of process capabilities ;- (i) a process capable of avoiding obstacles employing ultra-sonic sensors with a micro-switch detection ring as back up, (ii) a process which can wander around aimlessly (as a sort of "background" activity when the robot is doing nothing else), (iii) a "path recording" process which remembers the last 20 moves made by the robot and will allow it to back out of a blind alley when none of the other processes are doing anything constructive, and (iv) a light seeking or following process which enables the robot to follow a stationary or moving light or the robot could equally follow a bright stripe painted on the floor.

We have operated the robot with all four of these processes implemented. However further work is required in the area of prioritising of processes based on sensory input and decaying of priorities with time in order that the robot shows a stable behaviour or 'personality' as distinct from a skittish behaviour pattern.

As a simple example of subsumption in operation, depending on process priorities, the robot may follow a moving light until it encounters an obstacle. Due to sensory input the process priorities change and the obstacle avoidance process takes over, followed possibly by the wandering process until a point where the robot finds the light again. Such behaviour can be seen to display a certain primitive intelligence with some similarities to that of a bat or flying insect that feeds on glow worms, if indeed bats or other insects do feed on such worms and are not deterred by their dinner being lit. This question we will leave to the biologists.

As described above, subsumption could have possible application in manufacturing for both robot arms and mobile robots or AGVs. Subsumption control can improve the versatility of robots allowing them to operate in complex, unstructured environments. The task of perception and world modelling can be simplified and focused as necessary for each primitive process capability. Off-line programming could be simplified and proceed with out getting bogged down in kinematic modelling of the robot and it's environment. Will it happen ?

References

1. Brooks, R.A., A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, RA-2, 1986, pp 14-23.
2.. Flanagan, C., Toal, D. and Strunz, R., Subsumption Control of a Mobile Robot, Proc. Polymodel-16, Sunderland, 1995, pp 150-158.

Bibliography

1. Brooks, R.A., A robot that walks: emergent behaviour from a carefully evolved network, Neural Computation. 1(2): 1989, pp 253-262.
2. Ferrell, C., Robust agent control of an autonomous robot with many sensors and actuators, MIT Artificial Intelligence Laboratory Technical Report 1443, 1993.
3. Brooks, R.A., Elephants don't play chess, Robotics and Autonomous Systems 6, 1990, pp 3-15.
4. Brooks, R.A., Intelligence without reason, Proc. 1991 International Joint Conference on Artificial Intelligence, 1991, pp 569-595.
5. Brooks, R.A., Stein, L.A., Building brains for bodies. MIT Artificial Intelligence Laboratory Memo 1439, 1993.
6. Brooks, R.A. The behaviour language: user's guide, MIT Artificial Intelligence Laboratory Memo 1227, 1990