NEURAL NETWORK CONTROLLER FOR MOBILE ROBOT INSECT

R.J.Mitchell, D.A.Keating, C.Kambhampati
Department of Cybernetics, University of Reading, U.K.

ABSTRACT

Researchers in the Department of Cybernetics are interested in the development of intelligent machines which can interact with their environment; they should perceive their surroundings and move around them accordingly. To this end, some simple robotic `insects' have been built. Initially these machines were programmed with suitable actions, though recently an extra microprocessor circuit has been added, containing a neural network, which allows the insect to learn these actions. This paper describes the insects, the controller, its implementation, the results obtained and an overview of further work.

1. INTRODUCTION

There is much interest in the development of intelligent machines which can learn from their environment. Such machines should perceive their surroundings and react accordingly. Many workers in the field have used sensors for this purpose which require complicated algorithms and sophisticated computers. However, there are many simple organisms which survive using very simple sensors. Therefore it was decided to work with simple devices initially, from which more complicated systems could be developed. Thus several small mobile robot `insects' have been built which can move around an environment which they perceive using simple ultrasonic sensors. These are modular devices which have been designed such that extra sensors can be added easily.

Initially, the behaviour of these insects was programmed off-line and installed in a look-up table; the data from the sensors specifying the location in the table, and the information there determining the actions of the insect.

Recently, however, a simple microprocessor circuit has been added to the insect which contains a controller which allows the insect to learn suitable behaviour. This controller, which can be described in terms of a set of fuzzy automata or a modified Hopfield neural network, has been implemented on the insect and shown to work appropriately.

The next section introduces the insects; this is followed by a description of how their actions are programmed by humans; the next section outlines how the insects learn their own behaviour; and then its implementation on a microprocessor circuit is described.

2. CYBERNETICS' INSECTS

So far, eight simple mobile robotic `insects' and two larger devices have been produced, and the next generation insect has been designed whose sensors have been improved. These insects have been used in various teaching and research projects.

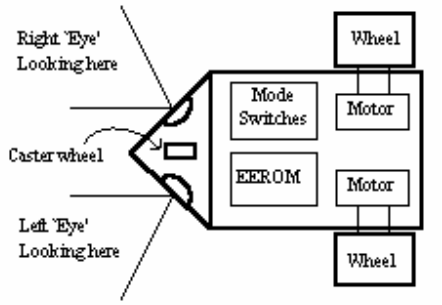

Figure 1. Bock Diagram of Robotic Insect

The simple insects have two ultrasonic sensors, which enable the insect to detect how far the nearest object is in front of a sensor, and two motors, each of which can be set to move forwards or backwards at a given speed. Figure 1 shows a block diagram of the insect.

The actions of the insect are determined using a look-up table, which is stored in an EEROM. The data from the ultrasonic sensors are passed to the address lines of the EEROM, and the value at the addressed location specifies the speed and direction of each motor. In fact, the insects can be programmed in various modes of operation which are selected by switches. The switch settings provide extra address lines to the EEROM.

The ultrasonic sensors are controlled by a programmable logic array (PLA) and associated analog electronics. This PLA causes an ultrasonic signal to be emitted by the transmitters for both eyes and then it waits for a signal to be reflected back from an obstacle to either receiver. The time taken before the reflection comes back (assuming one does return), which indicates the distance of the obstacle away from the nearest eye, is determined by the PLA. The PLA produces a one-bit signal indicating the eye closest to any obstacle and a further three bits indicating the distance of the obstacle from that eye. These three bits give a value in the range 0..7; if the value is 7 this means that no object is visible; due to limitations in the electronics controlling the ultrasonics, a value less than 4 means an obstacle is very close. This happens because the receiver cannot start to look for a return pulse too early, or it will pick up the transmit pulse.

These values, together with the values set by the mode switches, are passed to the address lines of the EEROM so as to specify a location in a look-up table which contains data specifying the velocity of the motors for each wheel for the current mode and sensor data. The velocity for each motor is specified by four bits, one bit indicates the direction, the other three specify a speed. The data at this address in the EEROM are passed to a second PLA which converts the information to PWM signals which are passed to MOSFET bridge circuits which drive the motors.

The insect can thus, for example, go forward or backwards at various speeds, or turn, by specifying that one motor goes forwards and the other goes backwards; and these movements are determined by the data from its sensors and the behaviour specified by the values in the look-up table.

Various improvements are to be made to the insect. A field programmable gate array (FPGA) has been designed to replace the two PLAs and provide extra facilities. For example, it will allow range values to be provided separately for each eye and it will provide a logarithmic measure of range rather than a linear one, thereby allowing greater accuracy when detecting close objects while still allowing the detection of distant objects. Improved electronics have also been designed to improve the range information of close objects. Also, it will be possible to connect the insect to a PC so that its EEROM can be programmed directly.

The insects are modular devices: it is easy to add extra sensors, for example, thermal infra-red sensors could be provided allowing the insects to follow a warm object like a human while avoiding cold objects like table legs.

The Department has also produced a larger version of the insect which has its own microprocessor and various extra sensors [1]. This has been programmed to move around avoiding obstacles, but it can also follow one infra-red source (a possible prey), avoid another (a possible predator), and `dock' with a charge unit (food) when it anticipates that its battery will soon start to run down. This behaviour has been built in to the unit by a specific program.

3. OFF-LINE PROGRAMMING OF THE INSECT

This section considers how the actions of the insect are programmed. As the insect does not have a microprocessor, the EEROM needs to be programmed off-line and then the EEROM inserted into the device. However, later versions of the insect will be programmable on-line as appropriate control circuitry has been programmed into the FPGA for this purpose.

Determining the behaviour of the insect has been set as a laboratory experiment in which students write four Modula-2 procedures determining four modes of behaviour for the insect. This is achieved by a program which generates an array, whose contents are specified by four procedures (Mode0, Mode1, Mode2 or Mode3; one for each mode) and which writes the array to a file. The data in this file are then programmed into the EEROM. The students write the procedures Mode0...Mode3; the following shows one such procedure which endeavours to make the insect move around avoiding obstacles.

```
PROCEDURE Mode0 (Range : INTEGER; LeftNotRight : BOOLEAN;
                 VAR LeftSpeed : INTEGER;  VAR RightSpeed : INTEGER);
(* Range is the distance of the closest object (0..7) from the eye specified by the
   boolean LeftNotRight; These values are used to set the speeds of the two motors
   in the range -4 (full backwards) to +4 (full forwards) *)
BEGIN
    IF Range = 7 THEN                (* nothing visible *)
        LeftSpeed := 4;   RightSpeed := 4;        (* so maximum speed forwards *)
    ELSIF Range <= 4 THEN            (* something too close *)
        LeftSpeed := -4;   RightSpeed := -4;      (* so go backwards *)
    ELSIF LeftNotRight THEN          (* something near left eye *)
        LeftSpeed := 4;   RightSpeed := -4;       (* so turn to the right *)
    ELSE                             (* something near right eye *)
        LeftSpeed := -4;   RightSpeed := 4;       (* so turn to the left *)
    END
END Mode0;
```

Other behaviours can be set by providing different rules, for example the insect can be programmed to follow objects; if it sees nothing it stays still, otherwise it moves towards the object but stops when it gets too close.

The above has been successfully used in laboratory classes. Students have produced various strategies to enable the insect to avoid obstacles, with varying degrees of success, but also with much ingenuity; there are many solutions to the problem. The insects have also been programmed to follow objects, and many insects have `waddled' in line, just like ducklings following their mother. `Puppy' mode has also been programmed, where the insect approaches the object, but backs away when it is too close to the object.

4. LEARNING STRATEGY

A strategy is required whereby the insect could learn these rules itself from its own experience. This strategy should be sufficiently simple that it could be incorporated in a dedicated FPGA or on a simple microprocessor. In this way the concept of a simple insect would be maintained; a single organism would start off having no in-built behaviour, but would subsequently move around its environment learning suitable actions.

It should be noted that it is not intended that the insect should learn the position of each obstacle in its environment and hence devise a path to move around the obstacles. Rather, the aim is for the insect to learn more general rules which allow it to investigate any environment without bumping into obstacles; the instincts of the insect are to keep moving and avoid obstacles, and the insect learns the actions needed to achieve these. When extra sensors are added to the insect, like the provision of `food', the instincts will also include the desire to find the food and the insect will learn the appropriate actions.

The strategy chosen to achieve these aims is based on a fuzzy automaton algorithm, though it has been modified suitably. As will be explained later, the strategy can also be considered as a modified Hopfield network. The basic idea, which is described in [2], and which has been used by [3 and 4], is as follows.

There are m possible actions, a1..am, associated with which are probabilities of taking these actions, p1..pm. The action which has the largest probability is usually used, but for the insect a weighted roulette wheel technique is used, whereby the action is chosen randomly with the action with the highest probability most likely to be chosen; this technique is used in genetic algorithms[5]. The performance of the chosen action is then evaluated as a success or a failure. As a result, the probabilities are changed according to the rules given below, where a is a performance value (positive meaning successful, negative meaning unsuccessful), n is the number of the action with the highest probability, and j is all values 1..m except for n.

```
IF  a >= 0 THEN  (*action successful *)
        pn = pn + a (1 - pn)      (* update probability of chosen action thus *)
        pj = pj (1 - a)           (* update all the other probabilities thus *)
ELSE
        pn = pn - a
        pj = pj + a / (m-1)
END
```

If an action is successful, its probability is increased and that of the others is decreased; but if the action is unsuccessful its probability is decreased and the others adjusted suitably. It

should be noted that, as one aim of this work is to implement the strategy on an actual robot insect in either an FPGA or a simple microprocessor, the probabilities are implemented as 8-bit positive integers rather than as fractions.

The above is not directly appropriate to the insect, as different strategies are required depending upon the position of obstacles relative to the insect. Therefore it was decided that one automaton would be used for each of the following five circumstances;

Case 1: no obstacle in front of an eye

Case 2: an obstacle relatively near the left eye

Case 3: an obstacle relatively near the right eye

Case 4: an obstacle in front of the left eye

Case 5: an obstacle in front of the right eye

The final system, therefore, is hierarchical; simple range information is used to choose one of five automata, the chosen one being used to select a suitable action and to learn from the results of that action.

Nine possible actions were chosen, where each motor could go forwards, stop, or go backwards (each motor is given a velocity of 4, 0 or -4); thus both motors could go forwards, one forward and one stopped, one forward and one backwards, etc.

The next problem was to decide whether a particular strategy was successful. This required careful thought. The aim was to produce simple `common sense' ways of evaluating the performance of an action, but ways which did not explicitly tell the insect what to do. These generate the performance value, a, which is used to update the probabilities, and which could be positive or negative. In general terms the rules to set a were as follows.

If no obstacle was visible then speed is important, the faster forward the higher the value of a. If an obstacle was very close to the insect, a is high if the action caused the insect to move away from the obstacle. If an obstacle is relatively close, then both factors are taken into consideration. These rules were encoded so as to generate a suitable a value, which could be positive or negative, with which the probabilities are updated.

The a factor is calculated by considering the speeds of the two motors, lspeed and rspeed, the ranges of obstacles from each eye, lrange and rrange, the previous values of these ranges, lrangewas and rrangewas, and the current automaton, case 1,.. case 5. The algorithm is:

```
IF case 1 THEN
        a = (lspeed + rspeed) / 4;
ELSE IF case 2 OR case 3 THEN
        a = (lspeed + rspeed) / 4 +  (lrange - lrangewas) + (rrange - rrangewas);
ELSE
        a = (lrange - lrangewas) + (rrange - rrangewas);
```

The main adaptive loop is thus as follows.

```
Choose One Probability Set (Automaton)
Choose Action, an, of this Automaton
Move Insect for I iterations
Evaluate Action and Update Probabilities
```

The strategy can be considered to be a modified Hopfield network[6] with modified Hebbian learning[7]. Figure 2a shows a Hopfield network with various inputs and outputs connected in a matrix with the junction between each input and output being a weight. Values are presented to the inputs and each output is the sum of the products of each weight on this line and its corresponding input. The largest output is the winner and if this is successful, the weights on its line are increased.

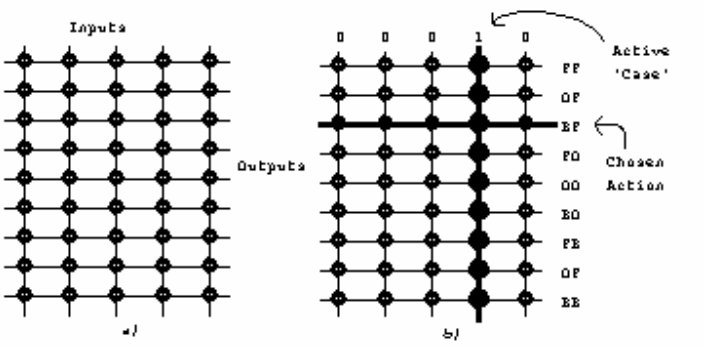

Figure 2. Strategy as a Hopfield Network

For the insect, there are five inputs (the five cases) and nine outputs (the nine actions); the inputs are binary and only one is a logic 1, corresponding to the current case, as is shown in figure 2b. The weights are the probabilities of the actions. The outputs are thus just the values of the probabilities of the current case. Rather than the highest output being chosen, the weighted roulette wheel system is used. Thus the strategy is a modified Hopfield network.

If the action is successful, the weight (or probability) of the chosen action for the current case is increased, and if the action is unsuccessful, this weight is decreased; all the other weights in the current case are adjusted accordingly (in effect they are all normalised). This can thus be considered to be a modified Hebbian learning scheme.

5. TESTS USING A SIMULATION

A simulation was written of an insect in a room with obstacles. This was used to test the algorithm before it could be implemented directly on the insect. The room is shown in figure 3; it has three obstacles on its walls and one in the middle. The figure shows the relative sizes of the insect, the ultrasonic beams it emits and the room.

The simulation allows various factors to be set. For example, the insect can use one set of probabilities or the five sets described above; also, it is possible to specify the number of times the insect is moved before the action is evaluated (I); also the value of a (which is first calculated in the range -3..3) is then transformed either linearly to values in the range -3ar, -2ar, -ar, 0, ar, 2ar, 3ar (for some constant ar), or more emphasis is given to very good and very bad actions, the values being -9ar, -4ar, -ar, 0, ar, 4ar, 9ar. It is also possible to `freeze' the probabilities after the simulation has been run for some time, and to fix the choice of action to the one with the highest associated probability. The most probable actions can then be encoded in a Mode procedure so as to verify that the insect has in fact learnt suitable actions.

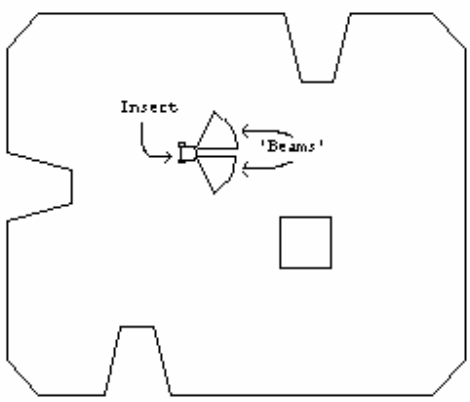

Figure 3. Simulation room used to test strategy

The simulation was run many times, with the random number generator starting with different values. The insect was initially placed in the middle of an environment where there were no near obstacles. In general it very quickly learnt to move with both motors forward or one forward and one stopped, but always it eventually learnt that both motors forward was the best strategy to adopt in open space. The insects also successfully learnt to turn away from obstacles to the left or to the right by turning one motor more than another.

One interesting result from the simulation is that the insect learnt different strategies depending upon its experience. Sometimes, for example, it learnt to turn away from obstacles which were relatively near, whereas at other times the insect would learn to turn only when it got very close to an obstacle. When avoiding an obstacle on the left, the insect sometimes learnt to turn the left motor forward and the right backward, or to turn the left forward and the right off, or even the left motor off and the right backwards; sometimes two or more of these options had a high probability, so the insect would use a combination of both actions.

The insect could also move successfully when only one probability set is used, but here, for example, it has to relearn how to avoid an obstacle on the left when it has not encountered one on the left for sometime.

Once the simulation has run for sometime and the updating of the probabilities is stopped, the insect successfully moves around its environment if the five sets of probabilities are used.

6. IMPLEMENTATION ON THE INSECT

The simulation demonstrated that the strategy works. The next stage was to implement it on the insect. To this end a simple microprocessor circuit was designed and built which plugged into the EEROM socket on the insect. The main requirement for this circuit was that it should consume little power, so a CMOS microprocessor was appropriate. Also, as the strategy was simple, an 8-bit microprocessor was sufficient. Thus, as a C compiler was available for the Z80 (and the simulation was written in C++ which could easily be turned into C), a CMOS Z80 microprocessor was selected. A block diagram of the circuit is shown in figure 4.

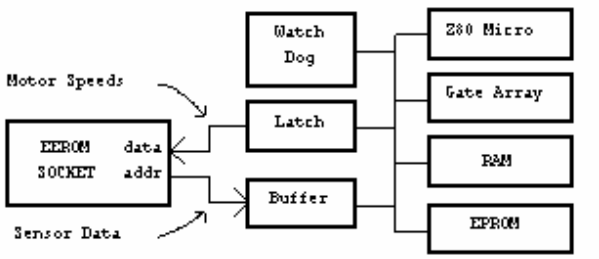

Figure 4. Block diagram of microprocessor circuit

The Z80 interfaces with the insect via a tristate buffer and a latch which respectively provide the data from the sensors (via the address lines of the EEROM socket) and generate the speed for the motors (using the data lines of the socket). The Z80 circuit also contains a 32K EPROM for the program and 32K RAM for data, a power-down circuit so the Z80 can enter low power mode when it has no other action, and an FPGA. This FPGA provides address decoding for the memories, buffer and latch (with expansion for more peripherals), a random number generator and (because there was space for it in the FPGA) a UART which could be used for communication with a computer, for monitoring the actions of the insect, or with other insects.

CONCLUSION

A strategy has been developed which allows simple robot insects to learn to move around an environment avoiding obstacles and this has been implemented in hardware on the insects. This, however, is just the beginning: further work is required to more fully investigate the strategy; to enable the insects to learn other tasks, like following objects; and then other sensors need to be added to the insect. It is also intended that the insects will be able to communicate with each other and report their actions to a logging computer.

REFERENCES

1. M. Hole and I. Kelly, "Robot Insect 2", Internal Report, Department of Cybernetics, University of Reading, June, 1993.
2. K. Narendra and M.A.L. Thathachar, "Learning Automata: an introduction", Prentice-Hall, 1989.
3. B.J. Oomen, N. Andrade and S.S. Iyengar, "Trajectory planning of robot manipulators in noisy workspace using stochastic automata", Int. Jnl of Robotics Research, 10, 135-138, 1991.
4. A. Tsoularis, C. Kambhampati and K. Warwick, "Trajectory planning of a robot using learning algorithms", Proc IEE Conf. Intelligent Systems Engineering, 13-16, 1992.
5. D.E. Goldberg, "Genetic Algorithms", Addison Wesley, 1989.
6. J.J. Hopfield, "Neural Networks and physical systems with emergent collective properties", Proc. Nat. Acad. Sci., USA 79, 2554-8, 1982.
7. D.E. Hebb, "The Organisation of Behaviour", Wiley, 1949.