IMPROVED LEARNING FOR A SIMPLE MOBILE ROBOT

R.J.Mitchell.

The University of Reading (UK)

ABSTRACT

The Department of Cybernetics has developed over the last few years some simple mobile robots that can explore an environment they perceive through simple ultrasonic sensors. Information from these sensors has allowed the robots to learn the simple task of moving around avoiding dynamic obstacles. This paper reports some new work done to further improve the learning process for these mobile robots.

Keywords: Learning and Adaptive Systems, Robotics

INTRODUCTION

There is much interest in the development of intelligent machines which can learn from their environment. Various machines have been produced which have many sensors, sometimes including high resolution video, which require a great deal of computing power to process the information coming in to the machine, and still more processing is then required to determine suitable action in response to this information.

Researchers in the Department of Cybernetics at the University of Reading believe that it is best to start with simpler systems. Also, we believe that there is much to learn from the behaviour of simple organisms like insects.

Therefore, a number of simple mobile robotic 'insects' have been developed which are small and which can operate rapidly. At Control ‘94 it was reported how these robots, equipped with two simple ultrasonic sensors, could learn how to explore its environment avoiding obstacles [MITCHELL 94a]. The concepts and the robots are briefly reviewed here.

The simple robots have two ultrasonic sensors, which enable the robot 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.

The information from the ultrasonic sensors is passed to suitable processing circuitry whose outputs specify at any instant the speed of the two motors. A block diagram of the basic robot is shown in figure 1.

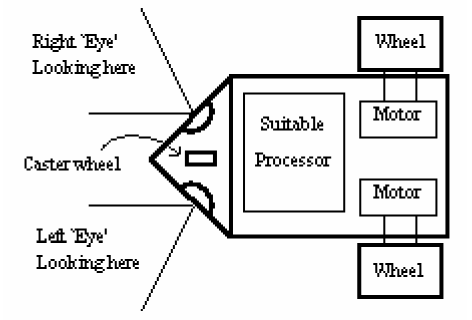


Figure 1. Block Diagram of Basic Robot

Initially the processing circuitry was a simple EEROM which contained a pre-programmed look-up table. The information from the sensors was passed to the address lines of the EEROM and the data at the specified address were the required motor speeds. The robot thus responded at any given instant to what its sensors detected, but its behaviour could not be learnt.

Then an extra Z80 based microprocessor circuit was added to the robots which implements a learning strategy based on fuzzy auomata [MITCHELL 94b]. This circuit takes information from the sensors, processing that information and driving the motors.

The microprocessor allows the robot to learn suitable behaviour, for example to move around an environment avoiding obstacles. This is achieved using a simple strategy, which is described briefly below.

This strategy involved an arbitrary choice by the designers of the algorithm, a choice which has been questioned. This paper reports some further experiments performed which investigate alternative choices which, it will be shown, improve the learning strategy. It is hoped that this work will then lead to further research which can remove the arbitrary choice.

This work complements other current research in the Department to improve learning by the use of many robots sharing their own experiences [KELLY, 98].

THE LEARNING STRATEGY

This section describes the algorithm which allows the robot to learn to move around its environment avoiding obstacles. The learning is achieved by a suitable program running on the Z80.

The technique used is, like the robots themselves, inspired by methods used in nature. When a baby is shown a toy it instinctively wants to grab the toy, but it does not know how. The baby is able to move its arms and legs but it is not aware which muscles control what part of the body. Initially, therefore, random movements occur. After a time, the baby stops moving its legs as it realises that the legs are not allowing the toy to be grabbed. A little later the baby learns to co-ordinate both hands, and finally is able to grab the toy. Thus the baby learns the appropriate action by trial and error: different actions are tried and the success or failure of the actions is used to determine the choice of later actions.

The robots require various actions, a means of choosing the actions, criteria for assessing the actions and a strategy for reassessing the choosing of the actions. The technique used to implement these ideas came originally from the concept of fuzzy automata [NARENDRA, 1989].

The basic idea is that the device has a number of possible actions and associated with each action is a probability of choosing that action. It is usual for the action with the highest probability to be chosen. Then the chosen action is performed, its success evaluated and this evaluation is used to adjust the probabilities: if the action was good then the probability of the action is increased and the probabilities of the other actions are decreased; if the action was poor, then its probability is decreased. Essentially therefore, good actions are rewarded and bad actions are penalised.

The robots are allowed various actions specifying how each of the two motors should move. Also, rather than always choosing the action with the highest probability, a ‘weighted roulette wheel’ technique is used - so the action with the highest probability is most likely to be chosen. One reason for using this technique is to try to stop the system getting trapped in local minima. Indeed, the probabilities can never be reduced to zero: there is therefore always a chance that each action is chosen at some stage.

Careful thought was required as regards determining whether the action is successful. The aim was to produce simple ‘common sense’ rules which did not directly ‘tell’ the robot how to behave. The rules chosen are as follows:

> If the robot was in the open, then going forward fast is good.
>
> If the robot was close to an obstacle, then moving away is good.
>
> If the robot was quite close, then a combination of these rules are used.

These rules are encoded to give a ‘goodness’ factor, α, which can be positive (good) or negative (bad). This factor is used to adjust the probabilities according to the algorithm given below. In this algorithm, m is the number of actions, n is the number of the action chosen, p_n is the probability of the chosen action, and p_j is the probability of the jth action (where j = 1..m, and j <> n).

```
IF $\alpha$ >= 0 THEN
    (* action was successful *)
  $p_n := p_n + \alpha (1 - p_n)$
    (* increase probability of chosen action *)
  $p_j := p_j (1 - \alpha)$
    (* decrease probability of other actions *)
ELSE
    (* action was unsuccessful *)
  $p_n := p_n - \alpha$
    (* decrease probability of chosen action *)
  $p_j := p_j + \alpha / (m-1)$
    (* increase probability of other actions *)
END
```

The possible actions are the state of each motor, each of which can be (B)ack, (F)orward or (O)ff. Thus there are nine actions:

FF	- both motors forward
FO	- left motor forward, right motor off
FB	- left motor forward, right motor backward
OF	- left motor off, right motor forward
OO	- both motors off
OB	- left motor off, right motor backward
BF	- left motor backward, right motor forward
BO	- left motor backward, right motor off
BB	- both motors backward

One action is chosen, based on the probabilities. The action with the highest probability is the one most likely to be chosen. The chosen action is evaluated and if it is good the probability of the action is increased, otherwise it is decreased. If the chosen action is good, then the association of the action with the input is reinforced.

One problem with this technique is that the action that is best for moving around when there is no obstacle near the robot is likely to be different from the action required to move away from the object.

Therefore five sets of the nine actions are used, each with its own set of probabilities, that is there are five sets of automata. These are activated under the following conditions: no obstacle visible; obstacle quite close on the left; obstacle quite close on the right; obstacle very close on the left and obstacle very close on the right. At any instance the current information from the sensors determines the one automata chosen.

The learning strategy therefore has two parts. First the current information from the sensors determines which set of actions and probabilities, that is which automata, to use. Then that automata is processed in the manner described above.

The success of the learning strategy has to be evaluated for useful experimentation to take place. This is done by the assignment of a suitable rating, for each action for each strategy[Kelly, 1997]. Each rating is in the range 4(good) to -4(bad). For instance, when the robot is out in the open, the choice FF is obviously the best, the choices FO or OF are quite good, etc. The following table is used:

Action	FF	FO	FB	OF	OO	OB	BF	BO	BB
Rating	4	2	0	2	0	-2	0	-2	-4

These rating values are then multiplied by the actual probability values of the actions in the current automata, to get a fitness value. Also calculated is the maximum possible fitness value (which occurs when the best action has the highest probability and all others have the lowest probability). The overall fitness for the automaton is then the actual fitness value divided by the maximum possible value.

Note, this fitness measure is used to evaluate the success of the learning strategy, but it is not used in the process to update the probabilities.

A criticism of this work is why are five sets of automata used? This seems arbitrary. Therefore research has begun into a method of automatically determining the number of automata. The first stage of this work, and that which is recorded here, is to investigate different numbers of automata.

CHANGING THE NUMBER OF AUTOMATA

The initial work used five automata, chosen according to the information from the sensors as to the distance of an obstacle from each eye. These are labelled:

DD - object distant for both eyes
?F - nearest object far from right eye
F? - nearest object far from left eye
?C - nearest object close to right eye
C? - nearest object close to left eye

The ? indicates that the distance of the object from the particular eye is not known, but that there is an object closer to the other eye. D is deemed to be further away than F which is further way from C. D is the maximum range of the ultrasonic sensors.

Different numbers of automata were then chosen, again based on the distance of objects from the eye. These are shown in Figure 2 below. The axes of the grids refer to the ranges of the (L)eft and (R)ight eyes, and the areas in which the grid is divided are labelled suitably. Note these areas include another range characterisation, I, meaning intermediate, which is between F and C. In all there are six sets, having respectively 4, 5, 7, 8, 10 and 16 sets of automata.

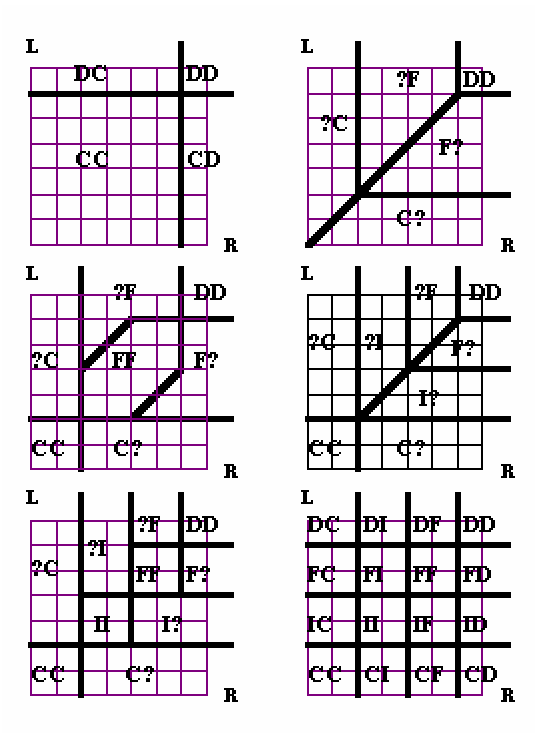


Figure 2. Different Sets of Automata

TESTS

Tests were performed, using simulation, on each of the six configurations. In each case the program ran for a set of length of time, from the same initial conditions, at the end of which the states of the probabilities for each set of actions was recorded, together with a measure of the success of what had been learnt, based on the fitness measurements.

A further test was then performed to see if the learning speed could be increased. The idea for this came from the learning strategy used in Kohonen networks [KOHONEN, 1984], where one node is adjusted by a certain amount, and then adjacent nodes are adjusted by a smaller amount. For the robot learning strategy, the normal algorithm was used to determine any changes to the probabilities of the chosen set of automata. In addition, those automata at ranges adjacent to the chosen set were also increased, but by half that amount. For instance, in the 8 probability case, when the chosen set was FF, then ?F and F? might be changed by the smaller amount. This concept can be justified easily, in that an action chosen when objects are at a certain distance will often be similar to the action chosen when the objects are only slightly further away. This is termed in the paper as ‘neighbourhood learning’.

Note: these tests have as yet been done in simulation only. The author intends that the program should be implemented on the Department’s robots, when time is available. However, in mitigation, it should be noted that the software used in the simulation contains the same code which was used to develop the work described in the Control 94 paper, and which was subsequently successfully transferred to the robots.

RESULTS

Two sets of results are here presented, first running the simulation for the six different sets of automata, and second, for eight probabilities, introducing the ‘neighbourhood learning’.

In each case the results are shown by a series of graphs depicting the probabilities for the particular automata. The graph is labelled by the ‘name’ of the set, below which is the fitness value and the number of times that set was chosen. The graphs show the values of the probabilities for each action at the end of the simulation.

For the neighbourhood learning test, below the sets of graphs is also shown the variation of fitness over time. This has two lines on it: one showing the fitness of the chosen action, one the overall fitness.

Figure 3 shows the state of the probabilities for each set of automata after the simulation has run. Figure 4 shows, for eight automata, the results of the simulation with and without neighbourhood learning. The graphs at the bottom of figure 4 show how the fitness values change with time. These graphs have two plots superimposed, one being the fitness of the action chosen at that time, the other being the fitness of all actions combined.

DISCUSSION

Looking at the graphs in figure 3, the overall fitnesses for the six conditions are as follows:

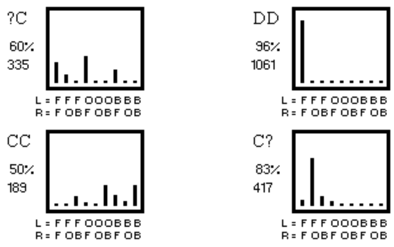
No. Automata	4	5	7	8	10	16
%Fitness	77	59	55	56	51	53

From this it might be concluded that the fewer the automata, the higher the fitness. However, in general, the automata with poor fitness values were those reflecting situations which were encountered infrequently. Referring to figure 3d) as an example, the lowest percentages were for states ?C, C? and CC, and these were encountered 33, 28 and 7 times, respectively: very little learning took place for those situations; whereas the DD state was encountered 1273 times. Also, in general, these automata were those in which the robot was close to an obstacle. Being close to an obstacle is not good given that the aim of the strategy is for obstacle avoidance. Therefore, having more automata seems to help the robot to avoid obstacles.

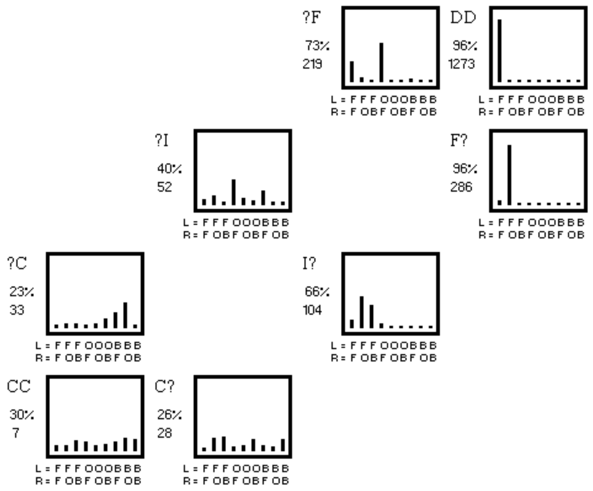
As regards figure 4, having ‘neighbourhood learning’ seems to improve the overall fitness (it is 40% in figure 4b and only 30% in figure 4a). In addition, as might be expected, neighbourhood learning means that the fitness curves rise more quickly. Neighbourhood learning, therefore, seems to improve the speed and performance of the learning strategy.

CONCLUSION

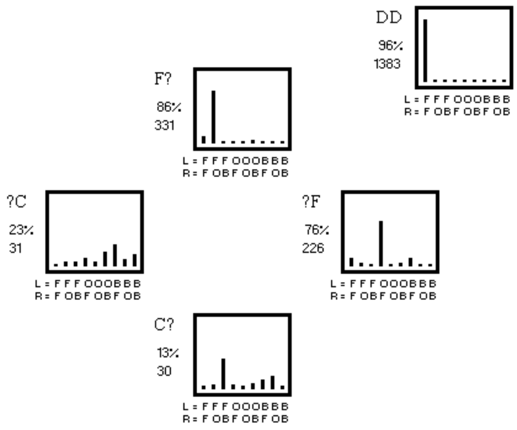
This paper describes some improvements to the learning strategy devised for the mobile robots. The extra sets of automata seem to keep the robots away from obstacles, and the use of neighbourhood learning speed the response of the learning. Future work includes further tests using the robots themselves, rather than simulation.



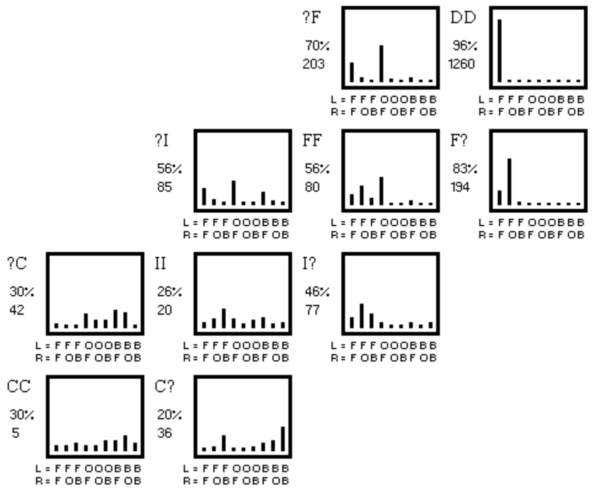
a) Four sets of automata



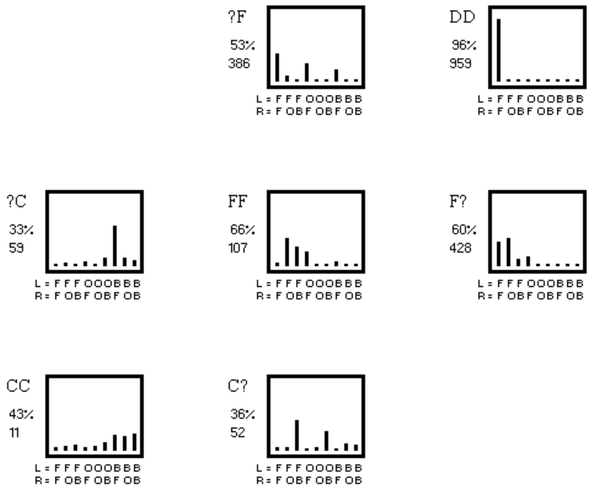
d) Eight sets of automata



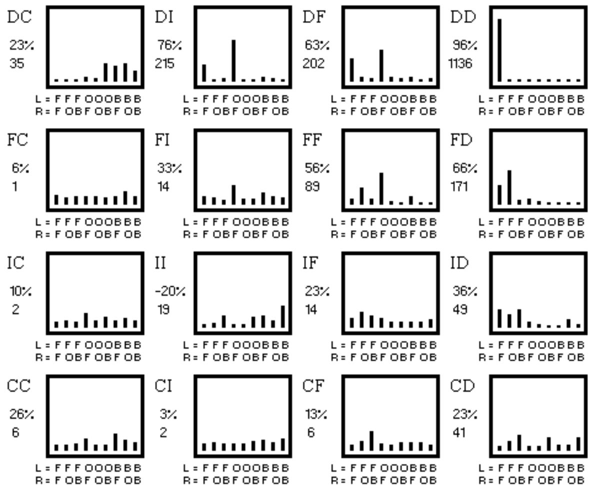
b) Five sets of automata



e) Ten sets of automata

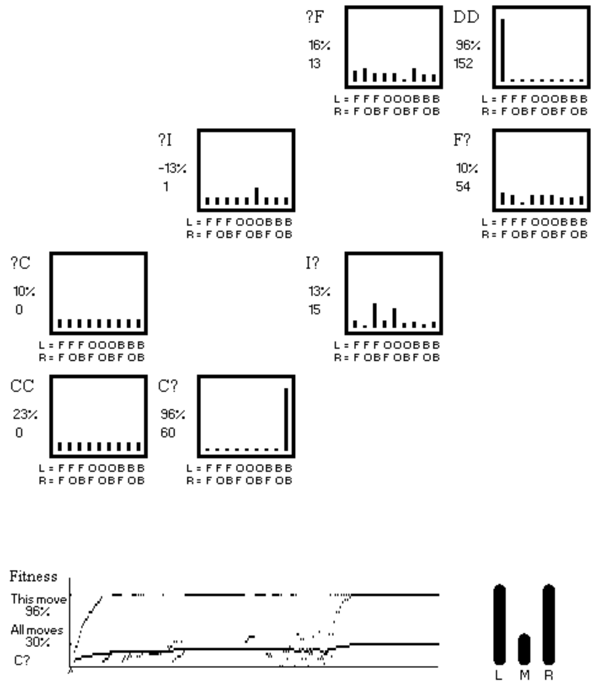


c) Seven sets of automata

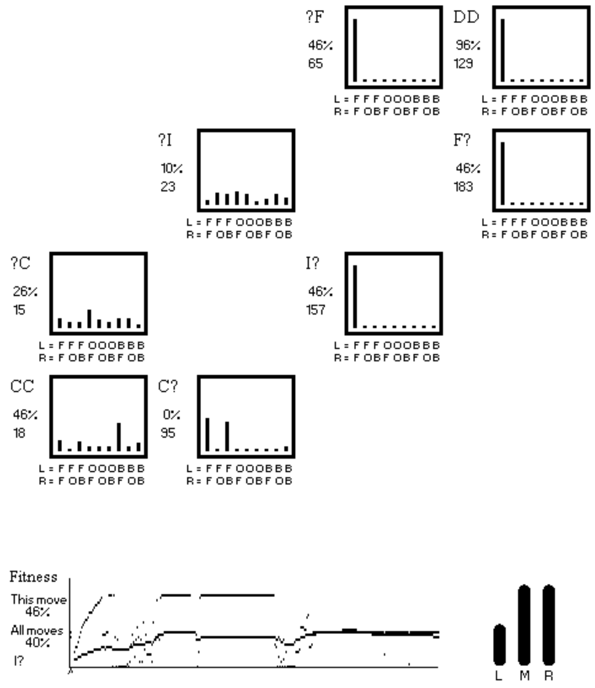


f) Sixteen sets of automata

Figure 3. Testing different number of automata



a) Without neighbourhood learning



b) With neighbourhood learning

Figure 4. Testing neighbourhood learning

REFERENCES

[KELLY 97] I.D. Kelly, The Development of Shared Experience Learning in a Group of Mobile Robots, PhD Thesis, The University of Reading.

[KELLY 98] I.D. Kelly and D.A. Keating, Increased learning rates through the sharing of experiences of multiple autonomous mobile robot agents, accepted for 1988 IEEE Int. Conf. of Fuzzy Systems, Alaska.

[KOHONEN, 1984] T. Kohonen, Self Organisation and Associative Memory', Springer Verlag.

[MITCHELL 94a] R.J. Mitchell, D.A. Keating and C. Kambhampati, (1994), Learning System for a Simple Robot Insect, Proc. Control '94, pp: 492-497.

[MITCHELL 94b] R.J. Mitchell, D.A. Keating and C. Kambhampati, 1994, Neural Network Controller for Mobile Robot Insect, Proc. EURISCON '94, pp: 78-85.

[NARENDRA, 1989] K. Narendra and M.A.L Thathacha, Learning Automata: an introduction, Prentice Hall, 1989.