Really Useful Robots[1]

U. Nehmzow, J. Hallam, T. Smithers
Department of Artificial Intelligence
Edinburgh University

Abstract

We propose a self-organizing type of controller for a mobile robot. Rather than decompose the control task into subtasks and implement these (an analytic approach) we propose to use *competence generators* which generate, through a process of self-organization, the necessary competences (a synthetic approach). We have built a mobile robot and equipped it with a first version of such a self-organizing controller.

1 Introduction

Controlling a mobile robot can be done in several ways: the *classical* approach is to equip the robot with control circuits, using feedback loops and traditional control mechanisms to achieve stable behaviour. This is an analytic approach: the control task is decomposed by the designer into subtasks which are then implemented, using standard control techniques.

Alternatively, there is the synthetic approach: generate the desired behaviour by combining basic competences, but using a 'bottom up' approach rather than a 'top down' one. In some sense Rodney Brooks [1,2] at MIT has chosen this approach by constructing robot controllers from simple extended finite state machines. The synthetic step here has taken place in the designer's mind.

We propose to take this approach one step further: is it possible to implement the synthesis step within the controller itself? We propose to use *competence generators* (which are driven by what we call instincts) that synthesize the competences necessary for the desired robot behaviour. This leads to the crucial questions 'What is the relationship between global behaviour and local action?' and 'Which combinations of instincts and generated competences instincts produce which local actions?'.

[1]Reprint from T. Kanade, F.C.A. Groen and L.O. Hertzberger (eds.), *Intelligent Autonomous Systems 2*, Proceedings of IAS 2, ISBN 90-800410-1-7, Amsterdam 1989.

In a spatial sense Luc Steels has worked on the first question. His *Reaction Diffusion Dynamics* [3] show intelligent overall behaviour (determining the shortest path between two points on a map, regardless of obstacles), although only local actions are defined (the physics of diffusion). However, for a robot controller, definition of spatial relationships alone is not sufficient: time and causality have to be considered as well.

2 Self-Organizing Systems

One promising approach for combining instincts and plasticity to generate competence is that of self-organizing systems. These have been proposed by various workers (e.g. Willshaw and von der Malsburg [4,5] to account for the autogenesis of topographic mappings in the nervous system). The system we have used shows properties of both Kohonen's self-organizing net [7] and Rumelhart, Hinton and Williams' back propagation network [6]. Therefore we will describe these systems first.

Kohonen's [7] self-organizing system consists of a two-dimensional array of cells which all receive the same input, the input vector $\vec{\imath}$. Each cell has an individual weight vector $\vec{w}$, the output o of each cell l is determined by the scalar product of input vector $\vec{\imath}$ and weight vector $\vec{w}$:

$$o_l = \sum_{j=1}^{n} w_{lj} \imath_j$$

where n is the number of input lines.

Self-organization takes place in the following way: after a stimulus is presented to the net, one cell is bound to respond most strongly to the particular stimulus, due to the fact that the initial weight vectors are randomly chosen. (All weight vectors are <u>unit</u> vectors, so the cell responding most strongly has a weight vector which is closest to the input vector.) The weights of this cell and of the cells in its neighbourhood are updated according to the following formula:

$$w_l(t_{k+1}) = w_l(t_k) + \eta(t_k)[\imath(t_k) - w_l(t_k)]$$

(η is the 'gain', initial value typically set at about 0.5). Weight vectors of cells outside that specified neighbourhood remain unchanged. After a number of input signal presentations and updates the net will show locally distant responses for different stimuli, i.e. a mapping has taken place.

This self-organizing system is used, for example, in Kohonen's 'Magic TV' (figure 1). The input image is a light spot which is detected by a simple camera. This camera consists of a round photocathode which is divided into three equal

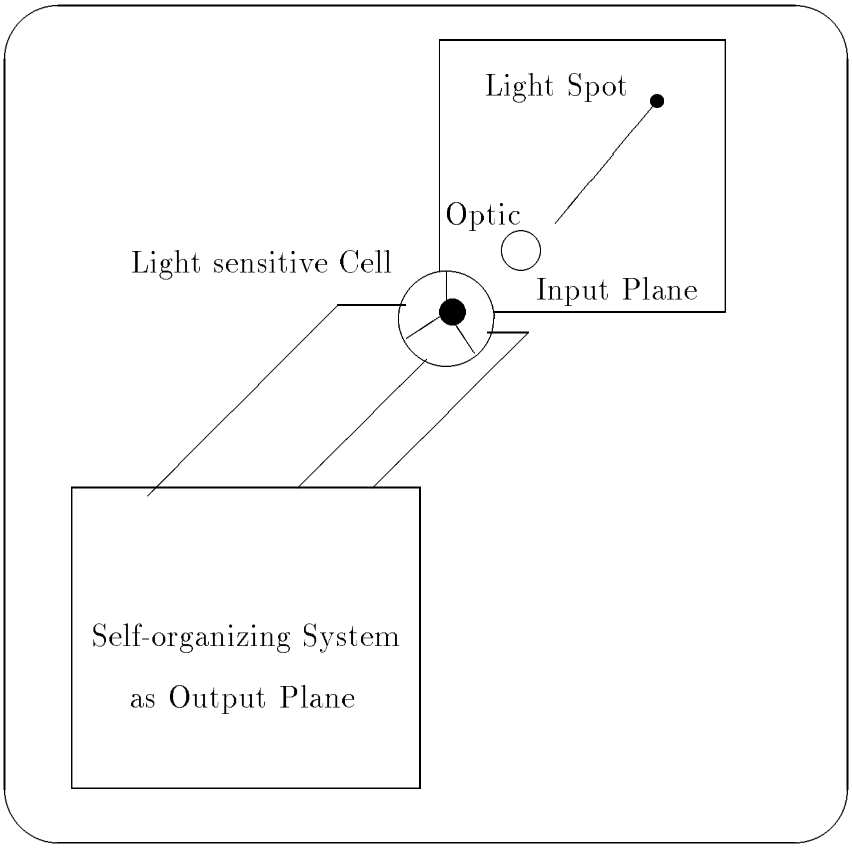

Figure 1: The 'Magic TV' (after Kohonen)

parts. Depending on the position of the light spot on the input plane, the three parts of the photocathode will receive different proportions of light and therefore emit different electrical signals.These signals are transferred to the self-organizing system as input signals. A number of different input signals are generated, the electrical signals are presented to the net, and the interesting observation is that after a while the position of the light spot on the input plane is reproduced on the net, i.e. cells that are in a similar position as the input light spot are most excited!

We define a <u>linear</u> system as a system where the output is a linear function of the input, similarly a <u>non-linear</u> system where the output is a non-linear function of the input. In this sense Kohonen's system is a linear system. Non-linear systems might also be interesting for our purposes (see section 4 of this paper).

The classical non-linear neural net, as described for example by Rumelhart, Hinton and Williams [6] (see figure 2), consists of a layer of input units, a layer of hidden units and one of output units. Input units simply pass the received signals on, the output of the hidden and the output units is determined by the following non-linear formula:

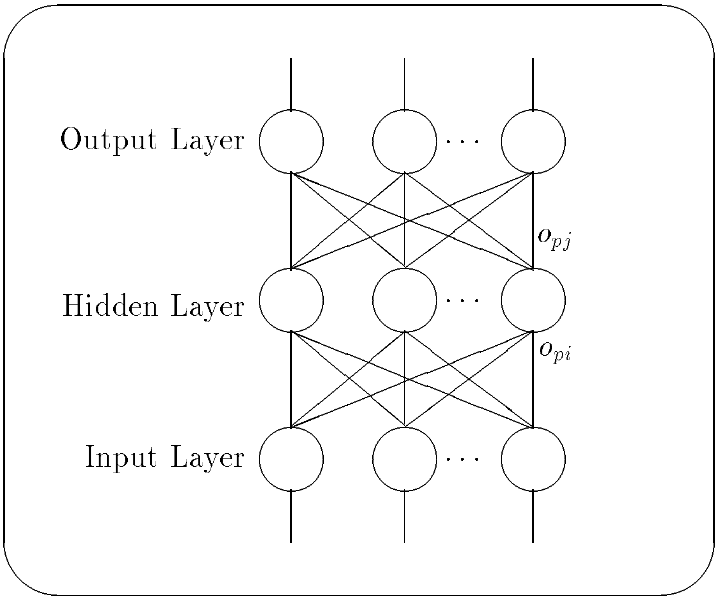

Figure 2: Non-linear Neural Net

$$o_{pj} = \frac{1}{1 + e^{-(\Theta_j + \sum_i o_{pi} w_{ji})}}$$

where Θ is a threshold value.

Such a net is not self-organizing, but capable of 'learning'. If the desired output vector $\vec{t}$ for a given input vector $\vec{\imath}$ is known, then learning can be achieved using back-propagation. The error signal for any output unit is determined by

$$\delta_{pj} = o_{pj}(1 - o_{pj})(t_{pj} - o_{pj}).$$

For hidden units the error signal is given by

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_{k=1}^{n} \delta_{pk} w_{kj}$$

where n is the number of output units.

Updating of weights is done according to the following formula:

$$\Delta w_{ji} = \eta \delta_{pj} o_{pi}$$

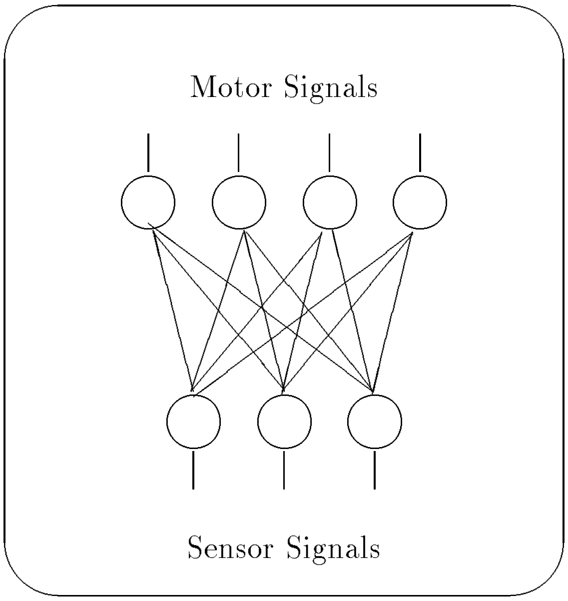

Figure 3: Linear Net as used in our Self-Organizing Controller

$$w_{ji}(t_{k+1}) = w_{ji}(t_k) + \Delta w_{ji}$$

(again η is the gain, o_{pi} the input to cell $\imath$.) Note that here the weight vectors are not unit vectors.

For our mobile robot we have used a linear system which shows properties of both approaches mentioned above. Our net consists of a number of input units (sensor signals) which are directly connected to a layer of output units (motor signals), see figure 3.

Again, input units simply pass the received signals on, the output o of the output units is determined by

$$o_j = \sum_i w_{ji} i_i$$

where $\vec{w}$ is the individual weight vector of each output unit. Note that the weight vectors $\vec{w}$ are unit vectors, as in Kohonen's system. If the desired output vector $\vec{t}$ to a given input vector $\vec{\imath}$ is known (this vector is determined by the robot itself, using a generate-and-test method), updating the weight vectors is done according to the following rule:

$$\Delta w_{ji} = \eta(t_j - o_j) i_\imath$$

$$w_{ji}(t_{k+1}) = w_{ji}(t_k) + \Delta w_{ji}$$

and $\vec{w}$ is renormalized after this operation.

This formula differs from the one used in Kohonen's self-organizing system in the way the input vector $\vec{\imath}$ is taken into account. The expression $\eta(t_j - o_j)i_i$ is proportional to $\frac{\partial E}{\partial w_{ji}}$ with $E = \sum_j (t_j - o_j)^2$.

3 Experimental Platform

Real-world situations can never be described completely by models. The real world often exhibits highly complex behaviour, therefore there will always be assumptions and simplifications that have to be made. For this reason simulation alone will not suffice. We are interested in robots interacting with the real world. Therefore experiments are very important. Our aim is to achieve true flexibility. The robot ought to be truly autonomous, independent from external knowledge as much as possible. The external knowledge we supply is reduced to some instincts which generate the basic behavioural patterns of the robot. We do not describe the task or the desired behaviour of the robot explicitly. The more general the task description is, the more information the robot has to find by itself, and the more independent it becomes.

As we said earlier: we want to look at a real robot, interacting with the real world, and being as independent from external knowledge as possible. We have therefore built such a robot, equipped with a self-organizing controller and some basic 'instincts', and watched it moving around in the real world. Hopefully our experiments will give us answers to the questions 'How is flexibility in unknown situations achieved' and 'What information is actually important for the robot, from the robot's point of view?'.

3.1 Preliminary Experiments

The mobile robot we use for our experiments is approximately 30 centimetres long, 25 centimetres high and 15 centimetres wide. Based on a Fischertechnik kit, the robot is equipped with two independent motors that can be used for driving and steering. The driving wheels are located almost in the middle of the robot so that it can pivot round its own axis. The self-organizing control algorithm is implemented using an ARC50 computer board. This is equipped with an INTEL 8052 CPU, 32k RAM and an EPROM Basic Interpreter. We have eight input and eight outputlines connected to a specially designed interface. This I/O Interface reads the sensory signals and switches the motors on or off, according to the signals from the computer board. Currently we are using tactile sensors only (whiskers): however, any sort of binary or even analog sensor could be used. Examples of this are infrared sensors or mercury switches. The power is supplied by two lead acid rechargeable batteries, giving us a working time of approximately sixty minutes. These batteries are heavy, therefore the vehicle can move only slowly (approximately $4cms^{-1}$).

3.2 The Computational Structure

To control this mobile robot we use a self-organizing controller. It consists of a memory unit and a critic (see figure 4).

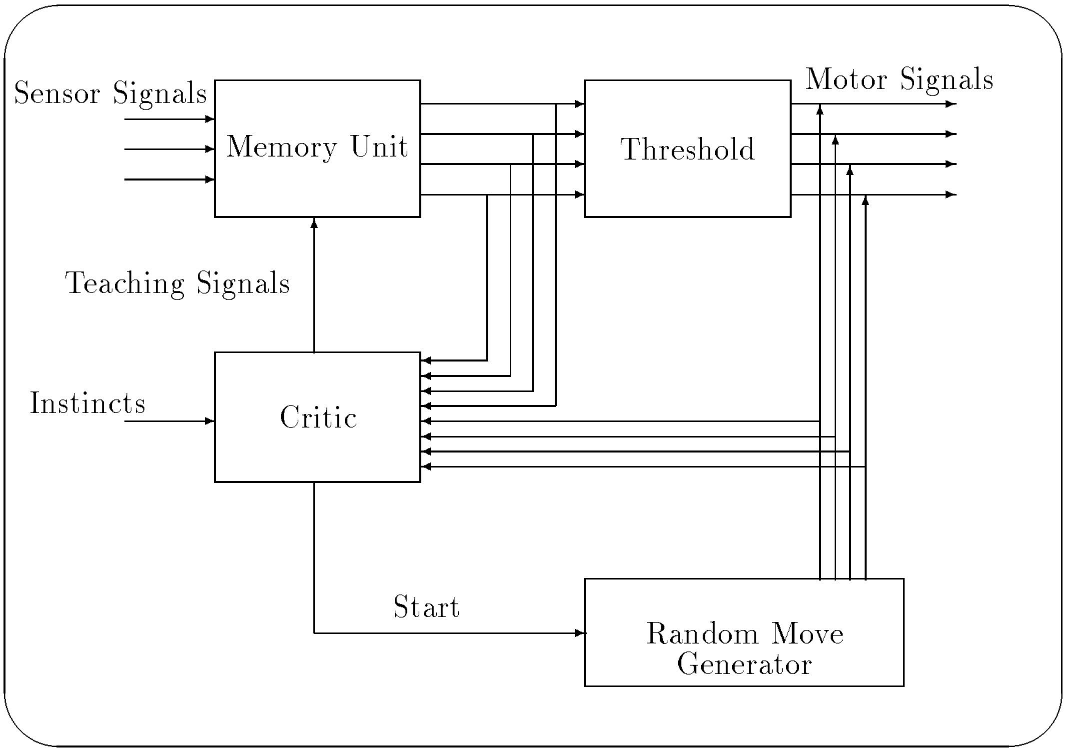

Figure 4: Computational Structure of the self-organizing Controller

The overall behaviour of the robot is influenced by two instincts:

1. Do not touch anything, and
2. Move forward, whenever possible.

These instincts are part of the *critic* which decides whether an action was successful (robot moved away from obstacle) or not (robot is still in contact with obstacle).

As long as no sensory signals are received ('no obstacle ahead') the second instinct will make the robot move forward. As soon as any of the whiskers detects an obstacle, however, this is no longer possible. The robot will therefore try the action that is stored in its memory unit for that particular constellation of input signals. This action will be performed for a while, typically about five seconds. If the action was successful (i.e. the robot has moved away from the obstacle) all is well and a forward move is again possible. If, however, the input signals still indicate an obstacle (and this might well be an obstacle other than the original one - the robot might have moved into more trouble) something else has to be tried. At this point a generate-and-test procedure is used, similar to the 'associative search' of Barto and Sutton [8]. A random move out of either the whole set of possible motor actions or a restricted set of motor actions is selected. This move is then performed for a period of time. After each unsuccessful attempt to move away from the obstacle this period of time is extended because we assume that unless the robot moves clear of the obstacle it might have moved into even deeper problems. If, by following this method, a successfull move has been found, this move is stored in the memory unit. The next time the same set of input signals is detected this newly learned move will be performed, and hopefully be successfull straightaway. Teaching the memory unit is another task of the critic. (We have used the following teaching method in our experiments: if a move that was recalled from memory was successful, that particular move is taught once again, to confirm it; if a newly found move is to be learned, it is taught three times. The version of the net we have used has three input units (for three sensors) and four output units. The gain η is 0.2 .)

Obviously with two motors that can move either forward, backward or be switched off there are nine possible robot actions. The chances of selecting a 'good' move out of this set are smaller than picking a 'good' move from a restricted set. Such a restricted set could comprise the following actions: forward, backward, swift left and swift right. However, by restricting the set of possible actions flexibility may be reduced as well, because moves are ruled out that might be useful under certain circumstances. So far we have used the full set of robot actions.

3.3 Experimental Results

When switched on the robot starts moving forward, as long as there is no obstacle in the way. It is thus obeying the second built-in instinct. As soon as an obstacle is detected, the experience of the robot comes in: the robot tries the learnt move. In the beginning this move is 'do nothing' (because the memory is empty), and consequently the robot sits still (which does not solve the problem). Because the obstacle is still touched, the robot now tries another, randomly chosen action out of the set of possible actions. If this action proves to be successful, the robot will store it in its internal memory, otherwise yet another action will be tried. We observed that if the robot is left for some time, trying out moves and learning the good ones, the robot eventually behaves very efficiently. Without hesitation it turns away from an obstacle, usually using the action that will achieve this the quickest way.

There are a few very interesting side observations. For example: it doesn't matter at all which way you plug in the sensors, or the motors (as long as sensors can be read and motors switched on or off, of course). Because the system is self-organizing it learns what to do, regardless of whether a motor is plugged in 'a particular way round' or not. The same applies to the sensors: whether a sensor is read on the most significant bit, or the least, or whether it is an infrared sensor or a whisker, does not matter. And it does not matter how many sensors are actually plugged in. The controller will make use of all the information available. Furthermore we observed that if a learnt reaction was no longer successful, for example because one of the motors had failed or because the polarity of the wires had been reversed, the control algorithm would find new solutions to comply with the built-in instincts.

4 Discussion

Networks - regardless of being linear or non-linear - have the advantage that the problem of increased complexity can easily be handled. Adding more sensors or output functions to a self-organizing controller simply means increasing the size of the net, but not changing the general structure of the controller. In the case of our robot, for example, it means that additional sensors are simply plugged into the net, the additional information is automatically taken into account by the system.

Although we have so far used a linear net, we intend to use a non-linear net as described above in the future. Non-linear nets have a higher signal-to-noise ratio than linear nets, which increases their reliability greatly. Due to this we can make use of another propertiy of nets: in networks (both linear and non-linear) associations between input signals and reactions emerge, even if they have not been explicitly taught. Experiments show that these emergent associations are in some significant relation to their input signals. In other words: some sort of learning without teaching takes place. The high signal-to-noise ratio of non-linear nets is beneficial for the interpretation of such a net output.

5 Conclusion

We designed a self-organizing controller for a mobile robot. *Competence generators*, governed by instincts, generate the necessary competences (a synthetic rather than an analytic approach). We have built a mobile robot which serves as a first experimental platform. The controller of this robot is equipped with a linear net as a memory unit and a critic which provides teaching signals for the memory.

References

[1] Rodney A. Brooks,*Achieving Artificial Intelligence through building Robots*, MIT AI Memo No. 899, May 1986

[2] Rodney A. Brooks,*A Robust Layered Control System for a Mobile Robot*, MIT AI Memo No. 864, September 1985

[3] Luc Steels, *Steps towards Common Sense* in: *Proceedings of ECAI '88*

[4] D. Willshaw and C. von der Malsburg,*How patterned neural connections can be set up by self-organization*, Proc. R. Soc. Lond. B, 194, 431-445

[5] R. Durbin and D. Willshaw, *An analogue approach to the travelling salesman problem*, Oper. Res. 21, 498-516

[6] D.E. Rumelhart, G.E.Hinton and R.J. Williams, *Learning internal Representations by Error Propagation*, in: David E. Rumelhart and James L. McClelland (eds.), *Parallel Distributed Processing*, MIT Press, Cambridge Mass., 1986

[7] Teuvo Kohonen, *Self-Organization and Associative Memory*, Springer, Heidelberg, New York 1988

[8] Andrew G. Barto, Richard S. Sutton and Charles W. Anderson, *Neuron-like adaptive elements that can solve difficult learning control problems*, in: *IEEE Trans. on systems, man and cybernetics*, Vol SMC-13, No. 5, September/October 1983, pp. 834 - 846